SONY

MSX 2

HANDLEIDING VOOR

MSX-BASIC Versie 2.0

HIT BIT

VOORWOORD

Dit boek biedt een inleiding in MSX2-BASIC. Het is gesplitst in twee delen— BASIC VOOR BEGINNERS en BASIC VOOR GEVORDERDEN.
Het deel BASIC VOOR BEGINNERS is geschreven voor hen die BASIC voor het eerst onder ogen krijgen. Het laat zien waar BASIC in feite om draait en het geeft aanwijzingen voor het schrijven van eenvoudige BASIC programma's. Door te oefenen met het schrijven van de programma's in het deel BASIC VOOR BEGINNERS zijn al de voornaamste BASIC bevelen vlot te leren.
Diegenen die al bekend zijn met MSX-BASIC kunnen het deel BASIC VOOR BEGINNERS overslaan en rechtstreeks door gaan naar het deel BASIC VOOR GEVORDERDEN. Of, als je nog geen programma's in MSX-BASIC geschreven hebt, maar al wel wat ervaring hebt opgedaan met een andere BASIC taal, dan is het ook mogelijk het deel BASIC VOOR BEGINNERS snel door te nemen om te zien wat de voornaamste bevelen in MSX-BASIC zijn, om vervolgens door te gaan naar het deel BASIC VOOR GEVORDERDEN.

MSX-BASIC Versie 2.0

In dit boek wordt uitgelegd hoe programma's in MSX-BASIC Versie 2.0 geschreven worden.
MSX-BASIC Versie 2.0 is een "krachtiger" versie van hetzelfde MSX-BASIC dat gebruik wordt in de Sony computers HB-55P, HB-75P/B, HB-101P en HB-201P. Alle bevelen en funkties waarover MSX-BASIC beschikt zijn ook opgenomen in MSX-BASIC Versie 2.0. Elk programma dat geschreven is in MSX-BASIC kan ook verwerkt worden met een computer die gebruik maakt van MSX-BASIC Versie 2.0.
Bevelen en funkties die alleen gebruikt worden in Versie 2.0 worden in dit boek aangeduid als behorend tot **MSX2-BASIC**. Bevelen en funkties die deel uitmaken van zowel MSX-BASIC als MSX2-BASIC worden eenvoudigweg samengevat onder de noemer **BASIC**.

MSX II wordt MSX I : POKE -1,170
-1,255

INHOUD

BASIC VOOR BEGINNERS

BASIC VOOR GEVORDERDEN

Hoofdstuk 1
Wat is een Programma?

Geef je computer eerst de volgende bevelen.

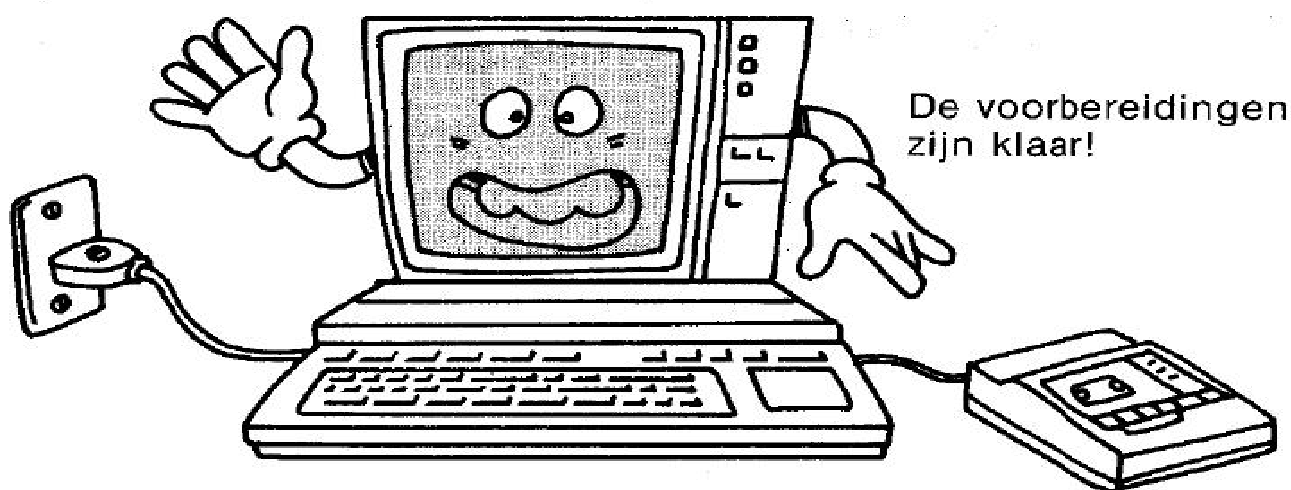
```
SET TITLE "",1
SET PROMPT "Ok"
SET ADJUST (0,0)
SET BEEP 1,3
SCREEN 0,,1,1,0,0
KEY ON
WIDTH 39
COLOR 15,4,4
```

JE EERSTE COMPUTER BEVELEN

- Hoe geef je de computer bevelen
- MSX-BASIC
- Berekeningen met PRINT "rekensom"
- Variabelen en het LET bevel
- Weergeven van woorden met PRINT "lettertekenrij"
- Lettertekenrij-variabelen

VOORBEREIDINGEN

Is alles in gereedheid, en ben je zelf ook klaar om BASIC programma's te leren schrijven? Zijn je computer en de TV of monitor aangesloten? Zo niet, kijk dan eerst in de gebruiksaanwijzing van de computer hoe dat moet. Als je computer niet beschikt over een ingebouwde diskette-eenheid, dan zul je een cassetterecorder of een losse diskette-eenheid moeten aansluiten. En daarmee zijn dan alle voorbereidingen getroffen.



Opmerking

Als je als beeldscherm een kleurenmonitor zonder ingebouwde luidspreker gebruikt, dan is het wel zaak ook nog een luidspreker aan te sluiten, want bij het bedienen van de computer speelt het geluid een belangrijke rol.

STARTEN VAN BASIC

Nu je alles in gereedheid hebt gebracht, kan je de computer inschakelen volgens de aanwijzingen in de gebruiksaanwijzing van de computer. De werkwijze voor het starten van BASIC is ook in de gebruiksaanwijzing beschreven. Na het starten van BASIC verschijnt op je beeldscherm één van de volgende twee beelden:

```
MSX BASIC version 2.0
Copyright 1985 by Microsoft
xxxxx Bytes free
Ok
■
```

Zonder diskette-eenheid

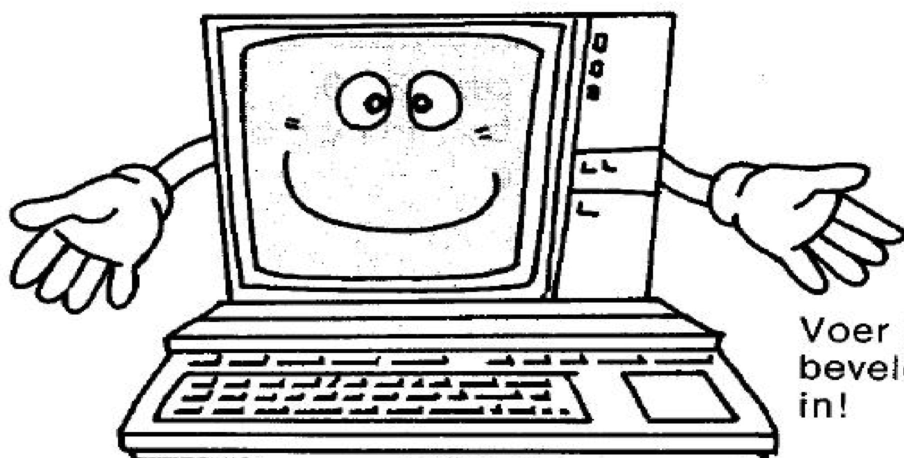
```
MSX BASIC version 2.0
Copyright 1985 by Microsoft
xxxxx Bytes free
Disk BASIC version 1.0
Ok
■
```

Met een diskette-eenheid

Als de laatste twee regels op het scherm "Ok" met daaronder een vierkant tekentje te zien geven, dan is BASIC klaar voor gebruik.

BEVELEN INVOEREN MET HET TOETSENBORD

Via het toetsenbord van je computer kan je de bevelen in de computer invoeren. Door het intypen van letters of nummers kan je de computer opdragen om een hele reeks uiteenlopende taken te verrichten.



Voer je bevelen hier in!

BEVEL 1—"KOFFIE ALSJEBLIEFT"

Ach ja, laten we de computer om een kop koffie vragen. Typ met het toetsenbord de volgende letters in.

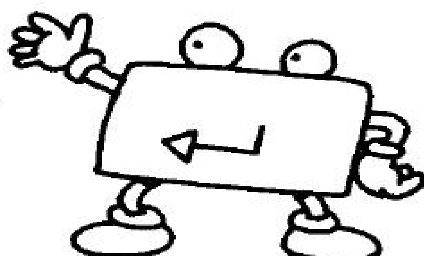
[K] [O] [F] [F] [I] [E] [   ] [A] [L] [S] [J] [E] [B] [L] [I] [E] [F] [T]

De lege [   ] ruimte in het midden is een spatie, in te voeren met de **spatiebalk** onderaan het toetsenbord. Bij het aanslaan van een lettertoets verschijnt het letterteken op de plaats van het vierkante blokje op je beeldscherm.
Dit blokje wordt de **cursor** genoemd. Het geeft steeds aan waar de volgende letter komt te staan.
Als je het verzoek "KOFFIE ALSJEBLIEFT" op de juiste wijze hebt ingevoerd, dan moet het beeldscherm er als volgt uitzien:

```
Ok
KOFFIE ALSJEBLIEFT■
```

De volgende stap is erg belangrijk. Na het invoeren van het bevel moet je de [↵] toets indrukken **(de RETURN toets)**. Door het indrukken hiervan laat je de computer weten dat het bevel kompleet is en dat de computer het zo moet uitvoeren.

KOFFIE ALSJEBLIEFT



"Dat is het hele bevel!"

Wat gebeurde er op het scherm na het indrukken van de [↵] toets?

```
Ok
KOFFIE ALSJEBLIEFT
Syntax error
█
```

Op het moment dat je de [↵] toets indrukt, hoor je een PIEP!-toon, waarna het bericht "Syntax error" op het scherm kwam te staan. Dit is het antwoord van de computer op je bevel, of verzoek, "KOFFIE ALSJEBLIEFT".
Het bericht "Syntax error" is een **foutmelding**. Het betekent dat er bij het invoeren van je bevel in de computer iets mis gegaan is. Je verzoek werd wel doorgegeven aan de computer, maar die begreep het niet.

BASIC
De computer is niet in staat een bevel als "KOFFIE ALSJEBLIEFT" te begrijpen, en daarom kreeg je ook geen koffie. Maar er zijn talloze andere bevelen die je computer wel begrijpt. Dit zijn de BASIC bevelen. Als je elk verzoek aan de computer nu maar in die BASIC bevelen uitdrukt, dan kan die vele interessante dingen voor je doen en zelfs knap moeilijke staaltjes verrichten. Laten we die bevelen die de computer begrijpt eens wat nader bekijken.

BEREKENINGEN PRINT uitdrukking

Op dezelfde manier als ons verzoek om koffie geven we de computer een volgend bevel, waarmee we hem opdragen 10 en 5 op te tellen. Voer de volgende lettertekens (letters, cijfers en symbolen) in:

[P] [R] [I] [N] [T] [ ] [1] [0] [+] [5] [↵]

Op het scherm verschijnt dan het volgende:

```
PRINT 10+5
 15
Ok
█
```

De computer heeft het PRINT 10 + 5 bevel opgevolgd, de rekensom uitgevoerd, en de uitkomst (15) op het scherm weergegeven.

PRINT rekensom

PRINT is het bevel dat de computer opdraagt alles wat er direkt op volgt op het scherm af te beelden. Dit is een van de BASIC bevelen, en de computer begrijpt het dus. Als je het bevel PRINT laat volgen door een rekensom, dan betekent dit voor de computer: "doe deze berekening en zet de uitkomst op het scherm".

Diverse berekeningen

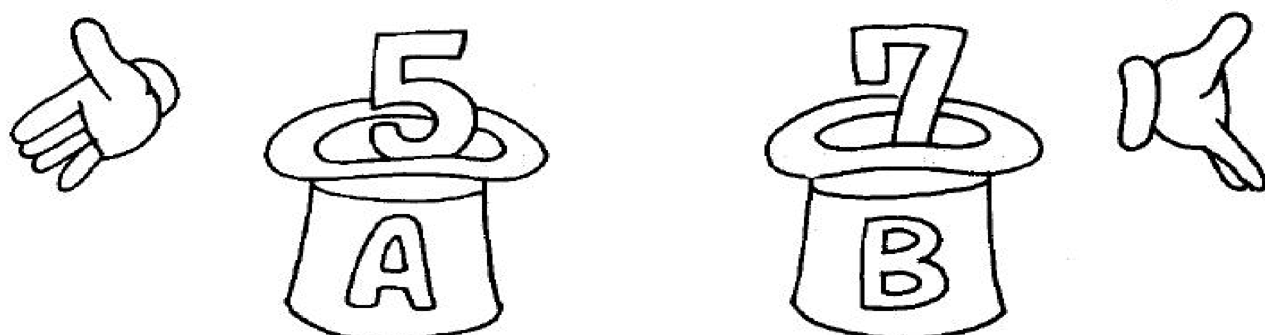
"10 + 5" is maar een simpele optelsom, maar de computer is ook in staat veel ingewikkelder berekeningen uit de voeren. De volgende rekenkundige symbolen maken deel uit van BASIC en zijn voor de computer te begrijpen.

	Symbool	Voorbeeld	Betekenis
Optellen	+	PRINT 123 + 234	123 + 234
Aftrekken	−	PRINT 300 − 125	300 − 125
Vermenigvuldigen	*	PRINT 9 * 8	9 × 8
Delen	/	PRINT 72/36	72 : 36
Machtsverheffen	∧	PRINT 5∧3	5^3

Als een van de rekenkundige bewerkingen in een berekening voorrang heeft, m.a.w. als je wilt dat die eerst verricht wordt, zet die bewerking dan tussen haakjes, ( ).

WAARDEN AAN VARIABELEN TOEWIJZEN LET

We stellen twee variabelen voor als goochelaarshoeden, en stoppen een 5 in de "A" hoed en een 7 in de "B" hoed.



Wat wordt A + B dan? Natuurlijk 5 + 7 = 12, nietwaar?
En we zien ook allemaal dat A × B gelijk is aan 35.

Leten we hetzelfde maar eens op de computer doen. Eerst stoppen we 5 in "A" en 7 in "B". Hiervoor typ je het volgende in:

L E T [ ] A = 5 ↵
L E T [ ] B = 7 ↵

```
LET A=5
Ok
LET B=7
Ok
■
```

De enige reaktie die de computer toont is weer "Ok", maar binnen de computer zijn zojuist de gekozen waarden aan de variabelen toegewezen, net als we de nummers in de hoge hoeden stopten.

LET variabele = waarde

LET is het bevel om een waarde aan een variabele toe te wijzen.

De A en B in de bovengenoemde bevelen kunnen we op dezelfde manier beschouwen als de toverhoeden in de plaatjes. Een letter zoals A of B die gebruik wordt om een getal te vertegenwoordigen heet een **variabele.**
Kijk nog een keer goed naar de vorm die het bovengenoemd toewijzingsbevel aanneemt.

```
LET A=5
```

De regel om te onthouden voor gebruik van het LET bevel luidt:

Zet de letter van de variabele links van het is-gelijk teken, en de getalswaarde rechts ervan.

Hierbij staat het = echter niet voor "is gelijk aan", zoals in de rekenkunde. De betekenis is veeleer "**het getal rechts gaat in de variabele links**".

We gaven de computer zojuist niet één, maar twee bevelen.

```
LET A=5
LET B=7
```

Daarmee is 5 in de variabele A gestopt, en 7 in B. Dit valt eenvoudig te kontroleren met het PRINT bevel.

```
PRINT A
 5
Ok
PRINT B
 7
Ok
```

Zo zie je, weer een voorbeeld van het "PRINT uitdrukking" bevel. Bovendien is het duidelijk dat je als onderdeel van de uitdrukking heel goed **variabelen** kan gebruiken. Laten we de volgende bevelen maar eens proberen:

```
PRINT A+B
 12
Ok
PRINT A*B
 35
Ok
```

Bij het invoeren van het "PRINT uitdrukking" bevel hebben we in de uitdrukking alleen de namen van de variabelen gebruikt, maar de computer gebruikte prompt de waarden daarvan, en gaf ons de uitkomst van beide sommen.

Een paar goocheltrucs met het LET bevel

```
LET C=A*2
OK
PRINT C
 10
OK
```

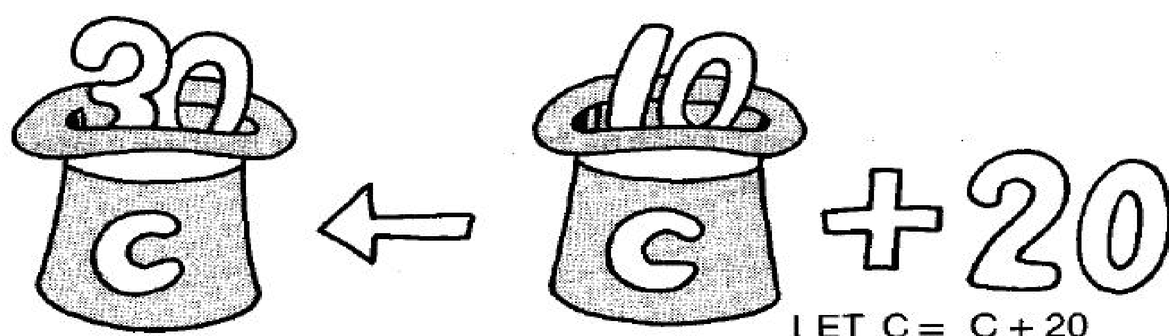
Hier is een nieuwe variabele C op het toneel verschenen. Met het **LET** bevel wijzen we aan C de waarde toe van de uitdrukking $A \times 2$. De variabele A was al bekend, net als de waarde ervan, 5. Zodoende werd de waarde van C gelijk aan 10.
Maar nu gebruiken we C nogmaals, om een nieuwe waarde te verkrijgen.

```
LET C=C+20
OK
PRINT C
 30
OK
```

Laten we het eerste bevel dat je de computer gaf eens bekijken:

```
LET C=C+20
```

Als je nu aanneemt dat = staat voor is-gelijk-aan, dan zal deze formule je wel bijzonder vreemd voorkomen. Denk er echter aan dat **het = teken de waarde van wat rechts ervan staat, toewijst aan wat links staat.** Aangezien de bestaande waarde van C 10 is, betekent de formule niets meer dan dat de computer deze waarde moet nemen en er 20 bij moet optellen, om C de nieuwe waarde van 30 te geven.



Misschien lijkt dit nu nog een wat lastige truc, maar als je hem eenmaal doorziet, dan is het eigenlijk heel simpel, en handig.

Tot op heden hebben we drie variabelen gebruikt—A, B en C. Vervolgens kunnen we een aantal algemene regels voor het gebruik van variabelen opstellen.

REGELS VOOR HET GEBRUIK VAN VARIABELEN

- In de naam van een variabele kun je zo veel letters gebruiken als je wilt, maar de computer leest alleen de eerste twee letters. Een aantal variabelen, genaamd
  ABC, ABCD, ABD, AB1, AB2
  worden door de computer beschouwd als slechts één en dezelfde variabele:
  AB
- Het is niet mogelijk voor het eerste letterteken van de naam van een variabele een cijfer, leesteken of ander symbool te gebruiken. De namen A1 of C3 zijn dus wel bruikbaar voor variabelen, maar namen als 1AB of *AB zijn niet toegestaan.
- Namen van funkties of bevelen in BASIC kunnen niet gebruikt worden als namen van variabelen. Ook niet als onderdeel ervan. De computer zal dergelijke namen niet accepteren. Termen als PRINT of LET, of ook namen als PRINTA of ALET, zijn dus niet toegestaan.

Weglaten van LET

Variabelen worden in BASIC veelvuldig gebruikt, en het LET bevel is daarom ook één van de meest gebruikte bevelen. Om de stroomlijn van een programma wat te verbeteren is het in BASIC daarom toegestaan bij het toewijzen van een waarde aan een variabele het woord LET zelf weg te laten.

In plaats van

LET A = 10

kun je dus ook eenvoudigweg schrijven

A = 10

Laten we nu maar eens wat andere bevelen maken met de "afgekorte" vorm van het LET bevel.

```
PRINT AB
 0
Ok
AB=100
Ok
CD=200
Ok
X=AB+CD
Ok
PRINT X
 300
Ok
PRINT ABC
 100
Ok
```

WEERGEVEN VAN LETTERS EN CIJFERS

PRINT "lettertekenrij"

Dit bevel dient om woorden op het beeldscherm weer te geven. Het is hetzelfde PRINT bevel dat we eerder zagen, maar nu gebruikt voor woorden. Probeer maar eens het volgende bevel.

[P] [R] [I] [N] [T] [ ] [“] [S] [O] [N] [Y] [”] [↵]

De aanhalingstekens (") zijn in te voeren door de wisseltoets ingedrukt te houden en de [2] toets aan te slaan.

```
PRINT "SONY"
SONY
Ok
```

Je kunt elk woord dat je wilt op het scherm zetten door het tussen aanhalingstekens, "   " op het PRINT bevel te laten volgen. Hierboven in het voorbeeld voerden we "SONY" in, waarna het woord zonder de aanhalingstekens op het scherm verscheen. Een woord dat er op deze wijze tussen aanhalingstekens in de computer invoeren noemen we een **lettertekenrij.**

PRINT "lettertekenrij"

Tussen de aanhalingstekens zijn alle mogelijke soorten "woorden" op het scherm af te beelden. Bijvoorbeeld:

```
PRINT "MSX2"
MSX2
Ok
PRINT "3+5"
3+5
Ok
```

In het tweede PRINT bevel

```
PRINT "3+5"
```

vormen de cijfers 3 + 5 een uitdrukking, een soort som, maar aangezien deze tussen aanhalingstekens staat, beschouwt de computer het slechts als "drie plus vijf", in plaats van een som die uitgerekend moet worden. Als je echter 3 + 5 invoert zonder de aanhalingstekens, dan slaat de computer wel aan het rekenen en verschijnt op het scherm de uitkomst 8. Bij het schrijven van computerprogramma's is het belangrijk te onthouden dat **getallen enerzijds beschouwd kunnen worden als numerieke waarden en anderzijds als lettertekens, afhankelijk van het gebruik of weglaten van aanhalingstekens**.

Letters kunnen ook opgeteld worden

Om letters op te tellen, d.w.z. samen te voegen, voer je ze tussen aanhalingstekens in, met een plus-teken er tussen. Probeer dit voorbeeld maar.

```
PRINT "SO"+"NY"
SONY
Ok
PRINT "MIJN"+"COMPUTER"
MIJN COMPUTER
Ok
```

In het eerste voorbeeld worden de lettertekenrijen "SO" en "NY" verbonden en samen weergegeven als het woord "SONY." Maar in het tweede voorbeeld is er na de N van "MIJN" een spatie gelaten. (Die spatie binnen de aanhalingstekens resulteert ook in een spatie tussen de woorden wanneer de computer de lettertekenrijen a.h.w. optelt en op het scherm zet. Zo zie je dat een spatie ook als een volwaardig letterteken meetelt.

spatie-teken

MIJN COMPUTER

lettertekenrij lettertekenrij

Nu we aldus lettertekenrijen kunnen maken en kombineren, is het tijd op zoek te gaan naar een soort toverhoeden (variabelen) om ze in te stoppen.

HET DOLLARTEKEN $ VOOR RIJ-VARIABELEN

Wat eerder stopten we getallen in variabelen als A, B en CD. Ook voor lettertekenrijen kunnen we variabelen gebruiken. Dit soort variabelen noemen we lettertekenrij-variabelen of kortweg **rij-variabelen**, om ze te onderscheiden van die variabelen die altijd getallen bevatten en die **numerieke variabelen** genoemd worden.
Om van een variabele een rij-variabele te maken hoeven we er slechts een dollar-teken aan te plakken—A$, B$ of CD$. De variabelen met een dollar-teken in de naam kunnen alleen gebruikt worden om een lettertekenrij te bevatten. (Als een getal zeg maar "letterlijk" als woord gebruikt wordt, kan ook dit aan een variabele met een $ toegewezen worden.)

Dan volgen hier een paar oefeningen met rij-variabelen.

```
LET A$="SONY"
Ok
PRINT A$
SONY
Ok
```

Hier gebruiken we het LET bevel om de lettertekenrij "SONY" als waarde toe te wijzen aan de rij-variabele A$. Bij het schrijven van het LET bevel hadden we het woord LET zelf ook weg kunnen laten. Wat niet vergeten mag worden zijn de aanhalingstekens, " " die nodig zijn om een lettertekenrij aan een rij-variabele toe te kunnen wijzen.

```
B$="123"
Ok
PRINT B$
123
Ok
C$=123
Type mismatch
```

In het bovenstaande voorbeeld trachtten we het getal 123 als waarde aan de lettertekenrij-variabele C$ toe te wijzen. Dit lukte niet, en in plaats van het bedoelde resultaat verscheen er een **"Type mismatch" foutmelding** op het scherm. Daardoor laat de computer je weten dat je appels en peren probeert op te tellen—twee dingen die niet samengaan—want een getal of numerieke waarde past niet in een rij-variabele.

FOUTMELDINGEN

Wanneer je de computer een onjuist bevel geeft, dan komt er op het scherm een mededeling te staan waarin de computer uitlegt wat er voor vergissing gemaakt is. Zo'n mededeling wordt een **"foutmelding"** genoemd. Als de computer niet in staat is de betekenis van een zojuist gegeven bevel te begrijpen, dan geeft hij

```
Syntax error
```

Of als je binnen een bevel waarden en variabelen die niet samengaan gebruikt, dan volgt

```
Type mismatch
```

zodat je weet wat er mis is.

MAAK EEN BASIC PROGRAMMA

- Een programma van één regel
- Kontroleer een programma met LIST
- Verwerk een programma met RUN
- Wis een programma uit het geheugen met NEW
- Waarden voor variabelen invoeren via het toetsenbord—INPUT
- Doe meer met het PRINT bevel
- Een deel van het programma wissen met DELETE
- Een beeld van het scherm wissen met CLS
- Maak een lus met GOTO

In het vorige hoofdstuk hebben we geleerd hoe je de computer een bevel geeft. Door het bevel op het toetsenbord te typen en vervolgens op de [↵] toets te drukken werd het bevel onmiddellijk uitgevoerd. Een computer is echter ontworpen om een hele reeks bevelen achter-een uit te voeren, terwijl je bij rechtstreekse invoer en onmiddellijke verwerking slechts één handeling tegelijk kunt uitvoeren. Er moet dus een andere manier zijn, en dat is waar het programma voor dient.

RECHTSTREEKSE VERWERKING EN PROGRAMMATISCHE VERWERKING

Invoeren van slechts één bevel tegelijk, en uitvoeren hiervan door in-drukken van de [↵] toets, zoals we tot op heden gedaan hebben, wordt **rechtstsreekse verwerking** genoemd. Er is nog een andere manier om de computer bevelen te laten verwerken, en dat is **programmatische verwerking**. Op de programmatische manier wordt de computer pas echt een volwaardige computer.

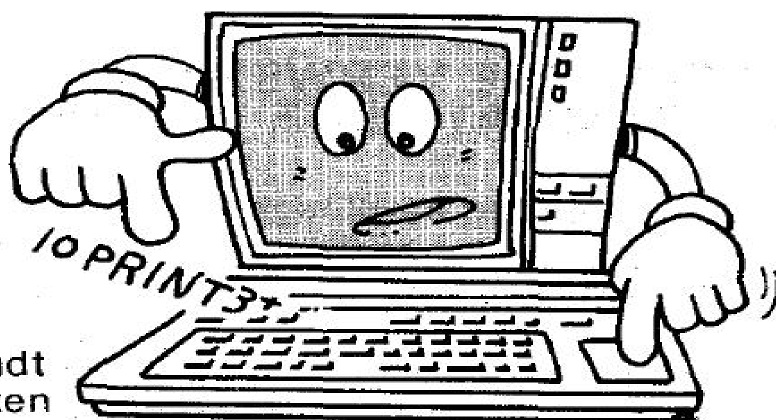
EEN PROGRAMMA VAN ÉÉN REGEL

We maken nu een programma op de programmatische manier. Dat is niets bijzonders op zich. Je voert alleen de volgende lettertekens in.

[1] [0] [ ] [P] [R] [I] [N] [T] [ ] [3] [+] [5] [↵]

```
10 PRINT 3+5
■
```

Het bevel PRINT 3 + 5 betekent, zoals we al eerder zagen, dat de computer de getallen 3 en 5 moet optellen en de uitkomst, 8, op het scherm moet weergeven. Maar toen je ter afsluiting van de invoer van het bevel de [↵] toets indrukte was er niets te zien. Dat komt omdat je voor het PRINT bevel het nummer 10 hebt ingevoerd. Door het invoeren van een nummer voor een bevel draag je de computer op om bij indrukken van de [↵] toets het bevel niet onmiddellijk uit te voeren, maar het in zijn geheugen op te slaan. Dit nummer wordt het **regelnummer** genoemd.



De computer onthoudt het bevel na indrukken van de [↵] toets.

Het bevel, voorafgegaan door een regelnummer, dat de computer onthield, was een **programma**.

KONTROLEER EEN PROGRAMMA MET LIST

De computer heeft het programma

```
10 PRINT 3+5
```

onthouden. Dit kunnen we kontroleren met behulp van

[LIST]

Voer via het toetsenbord de volgende letters in.
[L] [I] [S] [T] [↵]

```
LIST
10 PRINT 3+5
Ok
```

LIST is het bevel waarmee je de computer opdraagt de regels van een programma weer te geven.. Je voert dit bevel rechtstreeks in, zonder regelnummer ervoor.

VERWERK EEN PROGRAMMA MET RUN

Bij de rechtstreekse verwerking werd een bevel uitgevoerd door indrukken van de [↵] toets. Bij de programmatische verwerking gebruiken we hiervoor het RUN bevel.

[RUN]

Voer de volgende letters in.

[R] [U] [N] [↵]

```
RUN
 8
Ok
```

Het programma dat de computer onthouden had,

```
10 PRINT 3+5
```

wordt nu pas voor 't eerst uitgevoerd, en de uitkomst, 8, verschijnt op het scherm. Als je het RUN bevel nogmaals invoert, komt er weer 8 te staan. Je kan dit zo vaak doen als je wilt.

```
RUN
 8
Ok
RUN
 8
Ok
RUN
 8
Ok
```

Wat is een programma?
Laten we eens vergelijken hoe

```
PRINT 3+5
```

verschillend rechtstreeks en op de programmatische manier verwerkt wordt.

	Regelnummer	Rechtstreeks
Regelnummer	Nee	Ja
Voor uitvoeren	Druk op de [↵] toets	Geef het RUN bevel
Bevel onthouden?	Nee	Ja
Aantal malen uit te voeren	Slechts 1 maal	Zo vaak gewenst is

Omdat in een programma regelnummers gebruikt worden is de computer in staat het te onthouden. En het programma kan zo vaak worden uitgevoerd als maar gewenst is, met het RUN bevel.
Zonder programma's zou een computer nauwelijks een nuttig apparaat zijn. Een programma is een samenstel van een reeks opdrachten, die zo geschreven worden dat de computer een min of meer ingewikkeld stel handelingen achtereen uitvoert. De opdrachtenreeksen vallen onder de verzamelnaam **programmatuur**, in het Engels **software**.
De computer zelf is een machine die ontworpen is om op bevel een aantal handelingen uit te voeren. De computer is een stuk **apparatuur**, in het Engels **hardware**.

EEN PROGRAMMA UIT HET GEHEUGEN WISSEN NEW

We willen nu een nieuw programma schrijven en zorgen dat de computer het onthoudt, maar de computer heeft nog steeds het programma

```
10 PRINT 3+5
```

in het geheugen. Het is daarom nodig eerst het NEW bevel te geven, om te zorgen dat de computer het nieuwe programma in het geheugen kan opslaan.

NEW

Het NEW bevel wist het gehele programma, dat op dat moment onthouden wordt, uit het geheugen van de computer.

Voer het NEW bevel in, en geef dan het LIST bevel om te kontroleren of het programma inderdaad gewist is.

N E W ↵

```
NEW
Ok
LIST
Ok
```

En zoals je ziet verschijnt er geen programma, het is vergeten.

manieren om het geheugen te wissen

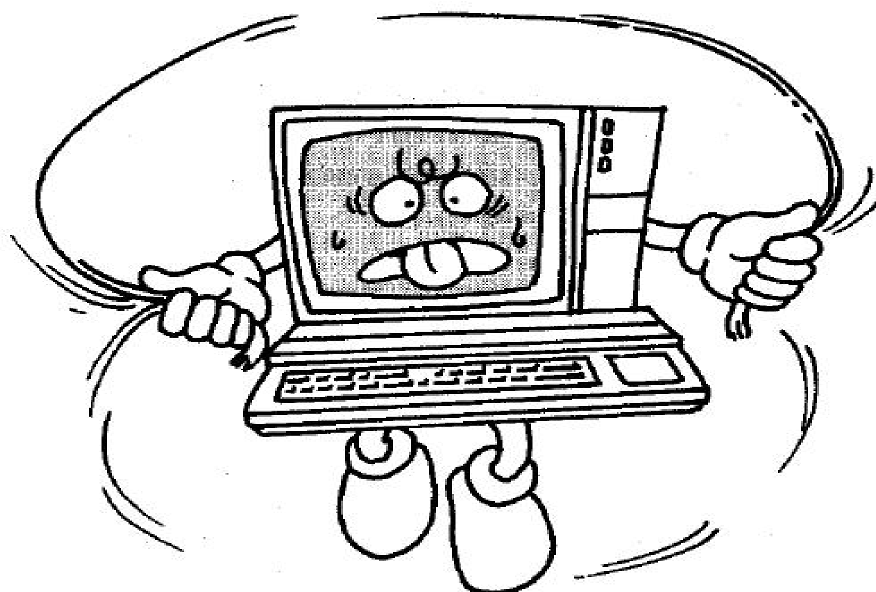
Je kunt een programma uit het geheugen van de computer wissen met behulp van het NEW bevel. Het programma wordt ook gewist door de volgende ingrepen.

- Indrukken van de [RESET] toets.
- Uitschakelen van de computer.

De [RESET] toets is bedoeld als een soort noodrem, voor als er iets mis is met het gemaakte programma, en het onmogelijk blijkt het programma te onderbreken met welke van de andere toetsen dan ook.

Indrukken van de [RESET] toets heeft hetzelfde effekt als uit- en weer inschakelen van de computer. Hierdoor wordt het programma in het geheugen van de computer gewist en komt de computer in dezelfde stand te staan als bij het voor 't eerst inschakelen.

Pas op dat je niet per ongeluk op de [RESET] toets of de aan/uit-schakelaar drukt terwijl je een programma aan het schrijven bent.





WAARDEN VOOR VARIABELEN INVOEREN VIA HET TOETSENBORD INPUT

Laten we nu een programma maken waarbij je bij iedere verwerking van het programma een andere waarde aan A en B kunt toewijzen. Bij iedere verwerking telt het programma dus verschillende getallen op. Dan moeten we echter in plaats van het LET bevel een ander bevel voor de variabelen gebruiken—en dat is het INPUT bevel.

INPUT naam variabele

INPUT is het bevel waarmee je de waarde van een variabele via het toetsenbord invoert.

```
10 INPUT A
20 INPUT B
30 PRINT A+B
```

Na het invoeren van deze regels kun je het programma zoals altijd weer kontroleren met het LIST bevel om te zien of alles juist onthouden wordt. Als de wijzigingen juist zijn aangebracht, kun je het programma gaan verwerken met het RUN bevel.

```
RUN
? ■
```

Bij het verwerken verschijnt onmiddellijk onder het RUN bevel een vraagteken, met daarnaast de cursor. Dit komt door het

```
INPUT A
```

bevel in regel 10. De computer wacht tot je hem via het toetsenbord laat weten welke waarde aan de variabele A moet worden toegewezen. Je kan nu elke gewenste waarde invoeren, maar laten we maar eens 25 proberen.

[2] [5] [↵]

Vergeet niet om na het typen van 25 weer op de [↵] toets te drukken, om de computer de opdracht te geven het getal 25 in de variabele A te stoppen.

```
RUN
? 25
? ■
```

Na het invoeren van 25 komt er weer een vraagteken te staan. Deze keer hoort het bij regel 20. Laten we aan variabele B de waarde 75 toewijzen.

[7] [5] [↵]

Nu je de waarde 25 hebt toegewezen aan A en 75 aan B, wordt regel 30 verwerkt en het totaal van A + B, 100, verschijnt op het scherm.

```
RUN
? 25
? 75
 100
Ok
```

Verwerk het programma nog eens, en voer andere waarden voor A en B in. Je zult zien dat telkens het juiste antwoord op het scherm afgebeeld wordt.

Maak het INPUT bevel meer "gebruiksvriendelijk"

Bij het verwerken van dit programma krijg je op het scherm eerst alleen maar een vraagteken en de cursor te zien. Als je het betreffende programma zelf geschreven hebt, dan weet je: dit betekent dat je de waarde voor A moet invoeren. Maar een ander heeft waarschijnlijk geen idee wat hij of zij met dat vraagteken aan moet. Om het programma meer "gebruiksvriendelijk" voor anderen te maken, kunnen we het INPUT bevel op de volgende manier aanpassen.

```
10 INPUT "A=";A
20 INPUT "B=";B
30 PRINT A+B
```

Verwerk het programma nog eens na het aanbrengen van de wijzigingen en wat zie je?

```
RUN
A=? ■
```

De lettertekenrij die tussen aanhalingstekens voor de naam van de variabele is gekomen heet een **oproep**. Bij het uitvoeren van het programma komt deze oproep voor het vraagteken op het scherm te staan. Gebruik van een oproep is handig om iedereen duidelijk te laten weten wat er ingevoerd moet worden.

INPUT "oproep"; naam variabele

Vergeet niet om tussen de oproep en de naam van de variabele een puntkomma (;) te zetten.

We verwerken het hele programma nog eens.

```
RUN
A=? 1234
B=? 4321
 5555
Ok
```

Voor de oproep kan je invullen wat je wilt. Bijvoorbeeld:

```
10 INPUT "Eerste getal";A
20 INPUT "Tweede getal";B
30 PRINT A+B
```

DOE MEER MET HET PRINT BEVEL

Door de wijzigingen die we in het laatste programma aangebracht hebben is de computer al wat meer in zijn rol gekomen. Maar er zijn nog heel wat andere ingrepen te verrichten om het bedieningsgemak te verbeteren en de scherm-aanduidingen wat duidelijker te maken. Zoals het programma nu is wordt alleen het totaal van A en B aangegeven, zonder dat er bij gezegd wordt wat het is. Het zou beter zijn als de computer ook aangaf dat het weergegeven getal de uitkomst van de som A + B is. Hiervoor kan je weer het PRINT bevel gebruiken om een regel aan het programma toe te voegen, als volgt:

```
10 INPUT "A=";A
20 INPUT "B=";B
25 PRINT "A+B="
30 PRINT A+B
```

We hebben aan het programma een regel 25 toegevoegd. Het is altijd mogelijk een extra regel aan een programma toe te voegen als er "Ok" op het scherm staat. Je typt dan de nieuwe regel op dezelfde manier als je een regel van een nieuw programma schrijft.

[2] [5] [ ] [P] [R] [I] [N] [T] [ ] [“] [A] [+] [B] [=] [”] [↵]

Geef daarna het LIST bevel om te kontroleren of regel 25 aan het programma is toegevoegd.

```
LIST
10 INPUT "A=";A
20 INPUT "B=";B
25 PRINT "A+B="
30 PRINT A+B
Ok
```

Regel 25 is tussen de regels 20 en 30 komen te staan, precies waar we hem wilden hebben. Het maakt niet uit in welke volgorde je de regels van een programma invoert. De computer zet ze automatisch in de juiste volgorde, van de laagst genummerde regel tot de hoogst ge-

nummerde. Bij het toevoegen van regel 25 heb je ook waarschijnlijk wel geraden waarom we voor de eerder geschreven regels de nummers 10, 20 en 30 gebruikt hebben. Dit liet ons ruimte genoeg om later extra regels, zoals de nieuwe regel 25, in te voegen.

Nu regel 25 onderdeel van het programma vormt zal de computer eerst "A + B = " op het scherm zetten voor het totaal van A en B verschijnt. Laten we het programma verwerken en eens kijken wat er op het scherm komt.

```
RUN
A=? 100
B=? 50
A+B= <-------- Het resultaat van regel 25
 150
```

Voor de uitkomst, die hier 150 is, komt er "A + B = " te staan, maar het zou nog beter zijn als zowel A + B = en 150 op een en dezelfde regel werden geschreven, dus zo:

```
A+B= 150
```

De puntkomma (;) om scherm-aanduidingen te verbinden
Als we regel 25 eens als volgt aanpassen:

```
25 PRINT "A+B=";
                ^-------- dit is nieuw
```

Na het laatste aanhalingsteken is een puntkomma (;) toegevoegd. Voor deze aanpassing volgen we een nieuwe procedure. Geef eerst het LIST bevel om regel 25 weer te geven.

```
LIST 25
```

Vergeet niet om na invoeren van het bevel weer op de [↵] toets te drukken.

```
LIST 25
25 PRINT "A+B="
Ok
■
```

LIST regelnummer

Je kan het LIST bevel gebruiken om een enkele regel van het programma op het scherm te zetten, door na LIST het regelnummer in te voeren. Ook kun je typen

LIST regelnummer—regelnummer

om een deel van het programma weer te geven. Bijvoorbeeld,

```
LIST 20-30
```

zal regels 20 t/m 30 van het programma weergeven. Voor ons programma zouden dat de regels 20, 25 en 30 zijn. Probeer het maar.

Na het weergeven van regel 25 met het LIST bevel kun je de cursortoetsen ( ) gebruiken om de cursor (het ■ teken) naar de plaats op het scherm direkt na het laatste letterteken (") van regel 25 te verplaatsen, want daar moet de puntkomma toegevoegd worden.

```
25 PRINT "A+B="■
Ok             ↑
               plaats de cursor hier
```

En druk de [;] toets in.

```
25 PRINT "A+B=";■
```

De puntkomma verschijnt dan op het scherm, maar hiermee zijn we er nog niet. Voor het doorvoeren van de wijziging moet je de [↵] toets indrukken. Bij het schrijven van het oorspronkelijke programma

heb je deze [↵] toets na elke regel ingedrukt. Net zo moet deze toets na een wijziging ook altijd ingedrukt worden. Pas door indrukken van de [↵] toets wordt de regel met wijziging en al in het geheugen van de computer opgeslagen.

Gebruik het LIST bevel om te kontroleren of regel 25 naar behoren aangepast is, en start dan de verwerking van het programma met het RUN bevel. Door het toevoegen van de puntkomma is de uitkomst van de optelling op dezelfde regel komen te staan als A + B = .

```
RUN
A=? 100
B=? 50
A+B= 150
OK
```

Gebruik van een komma (,) in plaats van een puntkomma (;)
Laten we in plaats van de puntkomma in regel 25 eens een komma zetten.

```
25 PRINT "A+B=",
```

↑ zet de komma hier

Om vervolgens, na het vervangen van de puntkomma door een komma, het programma te RUNnen.

```
RUN
A=? 100
B=? 50
A+B=            ␣150
```

Nu komt de uitkomst 16 spaties (de 15de spatie is voor het min-teken) rechts van het begin van de regel te staan, in plaats van vlak na A + B = .

EEN DEEL VAN HET PROGRAMMA WISSEN DELETE

De regels 25 en 30 van het programma luiden nu:

```
25 PRINT "A+B=",
30 PRINT A+B
```

We kunnen deze twee regels ook tot één regel samenvoegen.

```
25 PRINT "A+B=",A+B
```

Pas het programma aan door de twee regels te kombineren, met dezelfde procedure die we gebruikten toen we de puntkomma in een komma wijzigden.

Nu hebben we regel 30 niet meer nodig, en we kunnen deze uit het programma wissen. Er zijn twee manieren om een regel uit het programma te verwijderen. Een is met het DELETE bevel.

```
DELETE 30
```

Het DELETE bevel wist een aantal nader aangegeven regels uit een programma. Gewoonlijk wordt dit bevel als volgt geschreven:

DELETE regelnummer-regelnummer

Je typt DELETE met daarachter de eerste regel en de laatste regel van het te wissen deel van het programma, gescheiden door een streepje (-).
Als je echter maar één regel uit het programma wilt wissen is het niet nodig het DELETE bevel te gebruiken. Het volstaat dan om te typen

[3] [0] [↵]

EEN BEELD VAN HET SCHERM WISSEN CLS

Met de huidige vorm van het programma verschuiven telkens bij het verwerken de regels op het scherm, waarna de uitkomst wordt weergegeven. De regels die het resultaat vormden van de vorige verwerking blijven ook op het scherm staan. Het zou wellicht iets duidelijker zijn wat er precies gebeurt als het scherm telkens voor het verwerken van het programma werd schoongewist. Hiervoor nu hebben we het **CLS bevel**. (De letters CLS komen van de afkorting van het Engelse "clear screen".)

CLS

Laten we het CLS bevel maar aan ons programma toevoegen. We maken er regel 5 van.

```
LIST
5 CLS
10 INPUT "A=";A
20 INPUT "B=";B
25 PRINT "A+B=",A+B
```

Met deze regel 5 in het programma wordt het scherm telkens bij het uitvoeren van het programma gewist, hetgeen het gemakkelijker maakt om te zien wat er gebeurt.

MAAK EEN LUS GOTO

Lussen maken met het GOTO bevel

We voegen weer een nieuw bevel aan het programma toe.

```
30 GOTO 10
```

Vervolgens verwerken we het programma en kijken we wat er gebeurt.

GOTO regelnummer

GOTO draagt de computer op om door te gaan (of terug te gaan) naar het aangegeven regelnummer.

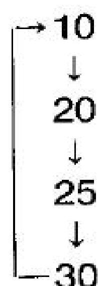
Gewoonlijk begint de verwerking van een programma bij de regel met het laagste regelnummer. Maar met het GOTO bevel kan je de loop van een programma aanpassen.

In dit geval stuurt het bevel

```
GOTO 10
```

de computer tijdens de verwerking terug naar regel nr. 10, en wordt het programma daar vandaan hervat. Altijd als je de computer op wilt dragen door te gaan met een ander deel van het programma, dan kan dat door het GOTO bevel te geven, samen met het gewenste regelnummer.

De reden waarom het programma blijft doorlopen is: nadat het GOTO 10 bevel in regel 30 de verwerking van het programma doet vervolgen met regel 10, worden eerst de regels 20 en 25 weer uitgevoerd, waarna het programma weer is aangeland bij regel 30, waardoor het weer teruggestuurd wordt naar regel 10, enzovoort, in een eindeloze cirkel. Het programma blijft in deze volgorde doorlopen:



De enige manier om deze cirkelgang te onderbreken is met CTRL + STOP.
Een deel van het programma dat op deze manier herhaald wordt heet een **lus**. En als die lus zonder ophouden maar herhaald blijft worden, zoals in ons programma, dan wordt het een **eindeloze lus** genoemd. Door het GOTO bevel hebben we dus een eindeloze lus in ons programma ingebouwd.

In dit hoofdstuk hebben we je laten kennismaken met het programma, en je ook een kijkje gegeven in de werking ervan. In het volgende hoofdstuk komen nog een aantal nieuwe bevelen aan bod, waarmee we gaan oefenen in het maken van wat meer geavanceerde programma's.

Hoofdstuk 2
Al wat meer een echte computer

ONTSNAPPEN UIT EEN LUS

- Voldoen aan een voorwaarde met IF—THEN
- Beëindigen van een programma—END
- Regels hernummeren met RENUM

VOLDOEN AAN EEN VOORWAARDE IF—THEN

De eindeloze lus

Wanneer een programma met een eindeloze lus wordt verwerkt met een RUN bevel, dan zal het gewoonlijk onophoudelijk door blijven lopen tot je het stopt, door gelijktijdig indrukken van de CTRL toets en de STOP toets. We zullen je hier echter nog een manier laten zien om uit een eindeloze lus te ontsnappen zonder gebruik van de CTRL + STOP toetsen. Bij de ontsnappingsmethode met de CTRL + STOP toetsen wordt het programma niet slechts onderbroken maar geheel afgebroken. Met de hier bescheven methode echter blijft het programma doorlopen nadat je uit de eindeloze lus bent ontsnapt.

Laten we eerst maar een eenvoudig programma met een eindeloze lus maken.

```
10  CLS
20  INPUT  "START";X
30  INPUT  "EEN  GETAL";Y
40  X=X+Y
50  PRINT  "TOTAAL=";X
60  PRINT
70  GOTO  30
```



Onderbreken van het programma aan de hand van de waarde van Y
Veel van de dingen die we doen op een doodgewone dag zijn afhankelijk van een aantal voorwaarden. Als het mooi weer is zal de sportdag misschien wel gehouden worden, maar als het regent, dan gaat het waarschijnlijk niet door. Als we geld hebben gaan we naar de film, maar zijn we platzak, dan blijven we thuis en gaan vroeg naar bed. Ontelbare malen komt het voor dat we het ene doen als aan een bepaalde voorwaarde voldaan is, maar iets anders als de voorwaarden het anders willen.
We kunnen de computer ook laten handelen op basis van zulke voorwaardelijke beslissingen. Het bevel dat we hebben om de computer de volgende handeling te laten verrichten indien aan een bepaalde voorwaarde wordt voldaan, is het **IF—THEN** bevel. Eigenlijk kunnen we een dergelijk bevel, dat de computer opdraagt een handeling te verrichten, ook een "**instruktie**" noemen, zodat we dus kunnen spreken van de "**IF—THEN instruktie.**" Dergelijke bevelen als **PRINT**, **LET** en **INPUT** dienen alle om de computer iets op te dragen, dus we kunnen het ook hebben over de **PRINT instruktie**, de **LET instruktie**, enz.

IF voorwaardelijke uitdrukking THEN instruktie

Met **IF—THEN** vertellen we de computer om, indien de voorwaardelijke uitdrukking die op **IF** volgt waar is, de instruktie die op **THEN** volgt uit te voeren.

Aan het programma dat we geschreven hebben voegen we nu het volgende **IF—THEN** bevel toe.

```
35 IF Y=0 THEN END
```

```
LIST
10 CLS
20 INPUT "START";X
30 INPUT "EEN GETAL";Y
35 IF Y=0 THEN END
40 X=X+Y
50 PRINT "TOTAAL=";X
60 PRINT
70 GOTO 30
```

De toegevoegde regel 35 betekent "als Y gelijk is aan 0, dan moet het programma beëindigd worden".

BEËINDIGEN VAN EEN PROGRAMMA END

In regel 35 luidt de voorwaardelijke uitdrukking Y = 0, en de instruktie is END.

END

END is het bevel dat dient om een programma te stoppen. Door het uitvoeren van dit bevel wordt het programma op dat punt beëindigd.

VOORWAARDELIJKE UITDRUKKINGEN

De onderstaande tabel verklaart de tekens die gebruikt kunnen worden om **voorwaardelijke uitdrukkingen** te schrijven.

Symbool	Betekenis	Voorbeeld	
=	Gelijk aan	IF A = B	Indien A gelijk is aan B
>	Groter dan	IF A>B	Indien A groter is dan B
<	Kleiner dan	IF A<B	Indien A kleiner is dan B
> =	Groter dan of	IF A> = B	Indien A groter dan of
= >	gelijk aan	IF A = >B	gelijk is aan B
< =	Kleiner dan of	IF A< = B	Indien A kleiner dan of
= <	gelijk aan	IF A = <B	gelijk is aan B
< >	Niet gelijk,	IF A< >B	Indien A niet gelijk is
> <	ongelijk aan	IF A> <B	aan B

Het teken dat we gebruikten in de voorwaardelijke uitdrukking van regel 35 is

```
IF Y=0
```

dus, "als Y gelijk is aan 0." Als aan deze voorwaarde wordt voldaan, dan vervolgt de computer met het uitvoeren van de instruktie die volgt na THEN. In ons programma is dit de END instruktie, en in dat geval is het programma dus ten einde.

Opgelet

In het LET bevel wordt het = teken gebruikt om aan te geven dat het getal rechts van het teken de waarde wordt van de variabele die ter linkerzijde met name genoemd wordt. Maar waar het = teken onderdeel vormt van een voorwaardelijke uitdrukking binnen een IF—THEN bevel, daar betekent het als vanouds "is gelijk aan."

We werpen een nog wat nadere blik op de werking en verwerking van het programma.

```
RUN
START? 10
EEN GETAL? 20
TOTAAL= 30

EEN GETAL? 15
TOTAAL= 45

EEN GETAL? 100
TOTAAL= 145

EEN GETAL? 0
OK
```

Eerst werd 20 bij 10 opgeteld, vervolgens werd bij het totaal 15 opgeteld, en toen werd daar weer 100 bij opgeteld. De eerstvolgende keer daarna dat een getal werd ingevoerd, was het 0. Op dat punt was de computer bij regel 30. Aangezien de ingevoerde 0 de waarde van variabele Y werd, was in regel 35 de voorwaardelijke uitdrukking Y = 0 waar, zodat de END instruktie volgend op THEN werd uitgevoerd, waarmee het programma afgelopen was.

REGELS HERNUMMEREN RENUM

Toen we begonnen het programma te schrijven nummerden we de regels in stappen van 10—10, 20, 30 ... 70. Later hebben we echter de regel 35 toegevoegd en daarmee de ordening in eenheden van 10 doorbroken. Nu is hier niets op tegen—als je wilt kun je voor de regels van je programma een volkomen onregelmatige nummering aanhouden, zoals 1, 3, 28, 29, 78, 105 ... enzovoort. Maar als je de regels liever in eenheden van 10 nummert, of als je misschien wel noodgedwongen zoveel regels tussen twee van de oorspronkelijke regels hebt moeten invoegen dat er geen ruimte meer over is, dan kun je de regels gemakkelijk hernummeren met behulp van het RENUM bevel.

We geven het bevel

```
RENUM
```

rechtstreeks, en kontroleren vervolgens of de regels nu netjes in stappen van 10 worden geschreven.

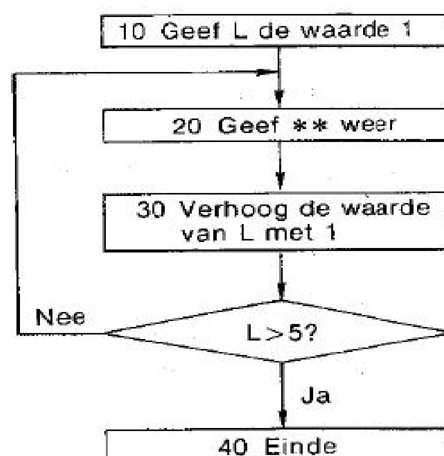
DE LUSSEN-SPECIALIST

- Herhalen van een lus met FOR—NEXT
- Een lus binnen een lus

HERHALEN VAN EEN LUS FOR—NEXT

We hebben geleerd hoe uit een lus te ontsnappen met het IF—THEN bevel.
Er is een manier om van tevoren het aantal malen te bepalen dat een lus doorlopen wordt; een manier die vaak wat prettiger in 't gebruik is. Deze nieuwe methode kunnen we het best leren door weer een eenvoudig programma te schrijven.

```
10 FOR L=1 TO 5
20 PRINT "** ";
30 NEXT L
```



Bij het verwerken van het programma komt er dit op het scherm.

```
RUN
** ** ** ** **
Ok
```

Het PRINT bevel in regel 20 van het programma geeft éénmaal ** ␣ weer. (␣ staat voor een spatie.)
In totaal wordt bij het uitvoeren van het gehele programma vijf maal ** ␣ weergegeven. Dit is omdat regel 20 vijf maal herhaald wordt. De opdracht voor het herhalen van regel 20 is gegeven in de instrukties van regels 10 en 30.

FOR variabele = beginwaarde TO eindwaarde
|
NEXT variabele

Het FOR—NEXT bevel zegt de computer dat deel van het programma dat zich tussen FOR en NEXT bevindt het aangegeven aantal malen te herhalen.

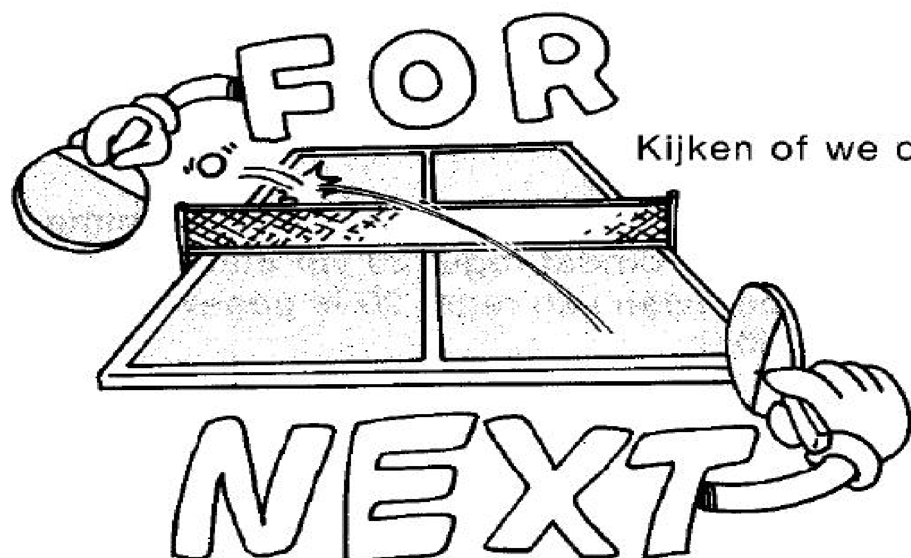
Regel 10,

```
FOR L=1 TO 5
```

en regel 30,

```
NEXT L
```

vormen een paar dat de computer opdraagt de waarde van L stap voor stap te verhogen, beginnend bij 1, en om het deel van het programma tussen de twee bevelen te herhalen tot L de waarde 5 bereikt. Bij verwerking van het programma krijgt L de waarde 1. Regel 10 geeft ook aan dat de eindwaarde van L 5 moet worden. Dan wordt regel 20 één maal uitgevoerd, waarna regel 30 de waarde van L met 1 verhoogt. Zolang L de waarde 5 nog niet bereikt heeft wordt de regel na het FOR bevel (regel 20) nogmaals verwerkt. Deze gang van zaken wordt herhaald tot de waarde van L gelijk is aan 5.
De FOR instruktie en de NEXT instruktie werken op deze wijze samen om het deel van het programma zo vaak te herhalen als je met de eindwaarde bepaald hebt. Aangezien ze altijd samen gebruikt worden, worden de twee instrukties samen aangeduid als het FOR—NEXT bevel, en een lus die hiermee gemaakt wordt heet een **FOR—NEXT lus**. Een belangrijk punt om te onthouden is dat **dezelfde variabele gebruikt moet worden** in de FOR instruktie en in de NEXT instruktie (in dit geval is dat L).



Een lus binnen een lus

Er zijn heel wat interessante dingen te doen met een FOR—NEXT bevel. Zo kan je binnen de FOR—NEXT lus die je met het eerste FOR—NEXT bevel hebt gemaakt, nog één of zelfs meer FOR—NEXT lussen opnemen.

De struktuur van het programma ziet er dan zo uit:

```
FOR L=1 TO 10
FOR M=1 TO 5
FOR N=1 TO 7



NEXT N
NEXT M
NEXT L
```

In dit voorbeeld bevindt lus (2) zich geheel binnen lus (1), en lus (3) bevindt zich helemaal middenin. Lus (1) wordt 10 keer herhaald, maar elke keer dat deze lus doorlopen wordt, wordt lus (2) daarbinnen vijf keer doorlopen. En voor elke herhaling van lus (2) wordt lus (3), daar weer binnen, zeven maal doorlopen.

Bij het schrijven van een dergelijk programma moet je voor elk van de lussen een andere variabele gebruiken (zoals L, M en N hierboven).

COMPUTERS EN TAAL

Een computer onthoudt je bevelen als een reeks opeenvolgende stappen, en voert ze uit in de volgorde waarin je ze geschreven en genummerd hebt. De termen waarin de reeks instrukties geschreven is wordt een **computer-taal** genoemd. BASIC is één van een aantal computer-talen.
Als je een programma in gewoon Nederlands zou kunnen schrijven, bijvoorbeeld zo:

1. Geef *** weer
2. Wacht even
3. Geef *** weer op de volgende regel

dan zou dat wel wat makkelijker zijn. Maar om te zorgen dat de computer bevelen die op die manier geschreven zijn ook kan begrijpen is een veel ingewikkelder soort programmatuur nodig dan de eenvoudige BASIC taal.

BASIC is echter één van de computer-talen die het dichtst in de buurt komt van een menselijke taal (Engels natuurlijk) en de termen zijn vrij makkelijk de begrijpen. **Maar aangezien het sterke punt van computers gelegen is in het omgaan met getallen en numerieke waarden**, in plaats van met gewone woorden zoals "Wacht even", is het in BASIC nodig om de computer een instruktie te geven die waarde van een variabele wijzigt, zoals

```
FOR T=1 TO 480:NEXT T
```

Hiermee bereik je hetzelfde resultaat, ook al is de computer niet in staat gewoon Nederlands te begrijpen.

Variabelen en regelnummers spelen een belangrijke rol bij het schrijven van computer-programma's. Oefenen om hier ervaring mee op te doen is daarom een van de sleutels tot succes bij het leren programmeren in BASIC.

GEGEVENS LEZEN

- Gegevens voorbereiden om ze als waarden toe te wijzen— READ—DATA

NOG EEN MANIER OM WAARDEN TOE TE WIJZEN READ-DATA

Zoals we gezien hebben speelt het gebruik van variabelen een hoofdrol bij het programmeren in BASIC. Allerlei soorten waarden worden aan variabelen toegewezen, waarna de laatste gebruikt worden voor het maken van berekeningen en beslissingen. Daarom is de methode van **toewijzen** van waarden aan variabelen ook van groot belang. Laten we de methoden waarmee we waarden toewijzen nog eens op een rijtje zetten.

Allereerst kunnen we waarden aan variabelen toewijzen met het LET bevel.
Om de waarde 100 aan variabele A toe te wijzen schrijven we:
LET A = 100 (of alleen A = 100).

De volgende methode die we geleerd hebben is met het INPUT bevel.
In tegenstelling tot het LET bevel, dat een vaste waarde aan een variabele in het programma toewijst, laat het INPUT bevel je tijdens de verwerking van het programma elke gewenste waarde via het toetsenbord invoeren.
Als je op het toetsenbord 500 typt, wanneer het bevel

```
INPUT A
```

wordt uitgevoerd en een vraagteken (?) verschijnt, dan wordt 500 als waarde aan de variabele A toegewezen.

Het READ-DATA bevel is nog een BASIC bevel voor het toewijzen van waarden aan variabelen.

We kunnen zien hoe het werkt met een eenvoudig programma.

```
10 READ A
20 PRINT A
30 DATA 185
RUN
 185
Ok
```

Wanneer regel 10

```
READ A
```

wordt uitgevoerd, dan wordt de waarde die gegeven is in de DATA instruktie (185, in regel 30 van het programma) aan variabele A toegewezen. (De waarde wordt op het scherm weergegeven door het PRINT bevel in regel 20.) Hetzelfde geldt ook voor lettertekenrij-variabelen.

```
10 READ B$
20 PRINT B$
30 DATA SONY
RUN
SONY
Ok
```

SONY is een lettertekenrij-waarde, maar in de DATA instruktie is het niet nodig de waarde tussen aanhalingstekens " " te zetten.
(Als je echter een komma binnen de lettertekenrij wilt gebruiken, of een spatie voor een woord wilt invoegen, dan moet de gehele lettertekenrij wel tussen " " gezet worden.)

```
10 READ B$
20 PRINT B$
30 DATA "SONY, HITBIT"
RUN
SONY, HITBIT
Ok
```

De hoeveelheid gegevens uitbreiden
De bovenstaande programma's wezen slechts één waarde aan slechts één variabele toe. Laten we het aantal gegevens (waarden) tot drie uitbreiden. Het is dan wel nodig ook het aantal variabelen tot drie uit te breiden.

```
10 READ A,B,C
20 PRINT A;B;C
30 DATA 10,20,30
RUN
 10  20  30
Ok
```

Voor het scheiden van de verschillende variabelen in het READ bevel worden komma's gebruikt. De gegevens in de DATA instruktie, die in dezelfde volgorde aan elk van de variabelen worden toegewezen, moeten ook door komma's gescheiden worden.

Het voornaamste punt om te onthouden is dat er tenminste zoveel gegevens beschikbaar moeten zijn als er variabelen in de READ bevelen aanwezig zijn.

PROGRAMMA'S OPSLAAN OP DE BAND

- Een cassetterecorder aansluiten
- Het programma op cassette opslaan met CSAVE
- Kontroleren of het programma juist is opgeslagen—CLOAD?
- Een programma vanaf cassette laden met CLOAD

Iedereen die wel eens een grote computer gezien heeft, is het waarschijnlijk opgevallen dat er een andere machine naast stond. Een machine waarin banden vooruit- en teruggespoeld worden en die er uitziet als een grote bandrecorder. In feite is een dergelijke machine ook een bandrecorder—één van de beste. Dergelijke bandrecorders worden gebruikt om de programma's, die een grote computer verwerkt, op te nemen.

Jouw eigen computer onthoudt een programma zolang de stroom ingeschakeld is. Wanneer je echter de computer uitschakelt, op de [RESET] (terugstel) toets drukt of het NEW bevel geeft, verdwijnt het programma onmiddellijk uit het geheugen van de computer.

Als je over een apparaat zou beschikken, zoals de bandrecorder die voor grote computers wordt gebruikt, dan zou je het programma op band kunnen opnemen (**opslaan**) alvorens je computer uit te schakelen. Ondanks het feit dat bij het uitschakelen het programma uit het geheugen van de computer wordt gewist, zou je bij opnieuw inschakelen in staat zijn het programma van de band in de computer te lezen (**laden**).

Gelukkig heb je niet zo'n grote bandrecorder nodig als zoéén die gebruikt wordt voor grote computers. Een normale cassetterecorder, die je misschien al bezit, voldoet ook prima als opnameapparaat voor jouw computer.

In dit gedeelte wordt uitgelegd hoe je een zelfgeschreven programma op een cassetteband kunt opslaan. Is jouw computer echter voorzien van een diskette-eenheid, dan kun je dit gedeelte overslaan en meteen verdergaan met het volgende gedeelte "Programma's opslaan op diskette". Hierin wordt beschreven hoe je programma's op diskette vastlegt.

Schakel je computer uit en ga te werk volgens de onderstaande aanwijzingen voor het aansluiten van een cassetterecorder op de computer, wanneer deze nog niet op elkaar aangesloten zijn. Bij uitschakelen van de computer wordt het programma uit het computergeheugen gewist. Voer daarom, na het aansluiten van de cassetterecorder, het programma opnieuw in.

Aansluiten van een cassetterecorder

Sluit je cassetterecorder met behulp van het aansluitsnoer van de cassetterecorder op de TAPE aansluiting van de computer aan. Je cassetterecorder is ook voorzien van een mikrofoonaansluiting (MIC) die je normaal gebruikt voor het opnemen van geluiden. Er bevindt zich tevens een oortelefoonaansluiting op de cassetterecorder, herkenbaar aan één van de volgende aanduidingen:

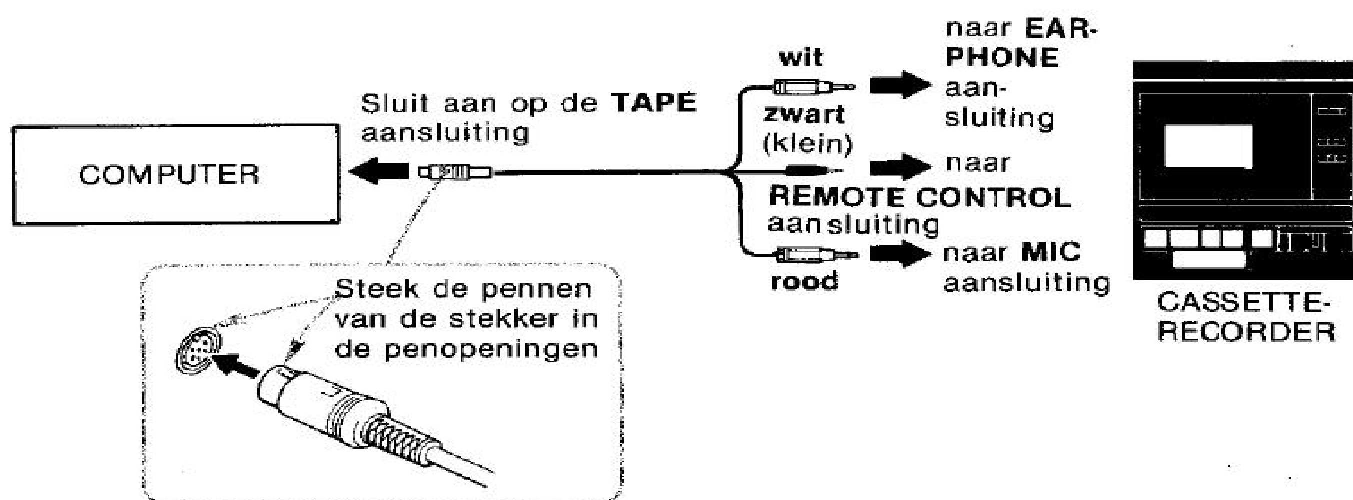
EAR, EARPHONE, MONITOR, MON, of ⑨

Het is ook mogelijk dat er een

REMOTE

aansluiting is voor afstandsbediening.

Verbind het aansluitsnoer van de computer met deze aansluitingen zoals in het onderstaande schema getoond wordt.



Mocht jouw cassetterecorder niet voorzien zijn van een afstandsbedieningsaansluiting, dan kun je het zwarte stekkertje los laten liggen.

HET PROGRAMMA OP CASSETTE OPSLAAN

CSAVE

Het BASIC bevel voor het opslaan van een computerprogramma op band is CSAVE.

CSAVE "bestandsnaam"

Door het geven van het CSAVE bevel wordt een programma met een bepaalde bestandsnaam in brugtaal op een cassettebandje opgenomen.

De bestandsnaam is de naam die je een programma geeft om het te kunnen onderscheiden van andere programma's.

Er zijn drie regels die je moet onthouden bij het geven van een bestandsnaam aan een programma:

- De naam kan uit maximaal 6 letters bestaan.
- Je kunt als aanvulling op de letters van het alfabet cijfers en tekens gebruiken.
- De computer maakt een onderscheid tussen kleine letters en hoofdletters.

Laten we een programma de bestandsnaam HART geven en het op cassetteband opslaan. Typ het volgende bevel.

Druk echter **niet** op de ↵ toets.

```
CSAVE "HART"
```

Zet nu, alvorens op de ↵ toets de drukken, de cassetterecorder in de opnamestand. Wanneer de computer met de afstandsbedieningsaansluiting is verbonden, zal de band niet beginnen te lopen. Er zal pas bij indrukken van de ↵ toets begonnen worden met opnemen. Het programma passeert via de TAPE aansluiting van de computer en wordt op de band opgenomen.

Wanneer er geen verbinding tussen de computer en de afstandsbedieningsaansluiting is gemaakt, wordt er begonnen met opnemen zodra de recorder in de opnamestand wordt gezet. Kontroleer of de band inderdaad loopt en druk meteen daarna op de ↵ toets.

KONTROLEREN OF HET PROGRAMMA JUIST IS OPGESLAGEN CLOAD?

Kontroleer altijd of het programma juist op de band is opgeslagen. Spoel eerst de band terug tot vlak voor het punt waar je bent begonnen met opnemen. (Bij sommige cassetterecorders moet je eerst de verbinding met de afstandsbedieningsaansluiting verbreken, voordat het mogelijk is de band terug te spoelen.) Zet vervolgens de volumeregelaar van de recorder in ongeveer de middelste stand en voer het volgende bevel in.

```
CLOAD? "HART"
```

CLOAD? "bestandnaam"

CLOAD? vergelijkt het programma in het computergeheugen en het programma op de band en kontroleert of deze inderdaad precies gelijk zijn.

Druk op de ↵ toets, na het CLOAD? bevel te hebben gegeven en zet de recorder in de **PLAY (weergave) stand.** (Als je op de PLAY toets (voor weergave) drukt terwijl de afstandsbedieningsaansluiting met de computer is verbonden, zal de band nog niet gaan lopen. De band gaat pas lopen nadat je de ↵ toets hebt ingedrukt.) Wanneer de band het punt bereikt waar het programma is opgenomen, dan verschijnt

```
Found:HART
```

op het scherm. Met deze mededeling vertelt de computer je dat deze het bestand met de naam "HART" op de band heeft gevonden. Vervolgens vergelijkt de computer het programma op de band met het programma dat in het geheugen opgeslagen is gebleven. De tijd die nodig is voor het vergelijken is afhankelijk van de lengte van het programma. Wanneer beide programma's exakt gelijk zijn verschijnt

```
OK
```

op het scherm.

Wanneer het programma op de band en dat in het geheugen niet met elkaar overeenstemmen, verschijnt

```
Verify error
```

op het scherm. Het "Verify error" bericht geeft aan dat het programma niet juist op de band is opgenomen. Bij het verschijnen van dit bericht moet je het programma opnieuw opslaan met het CSAVE bevel.

Het komt ook zo nu en dan voor dat het

```
Found:HART
```

bericht niet op het scherm verschijnt, ondanks het feit dat de band het punt waar begonnen werd met opnemen van het programma, gepasseerd is. Dit is meestal te wijten aan het feit dat het volume van de recorder te hoog of te laag is ingesteld. Stel het volume juist in en voer opnieuw het CLOAD? bevel uit. Kontroleer bij het verschijnen van het "Found: HART" bericht op het scherm de volume-instelling van de recorder, zodat je voortaan weet hoe het volume ingesteld moet zijn voor het CLOAD? bevel.

EEN PROGRAMMA VANAF CASSETTE LADEN
CLOAD

Het **CLOAD bevel** wordt gebruikt om de computer het programma vanaf de band in het computergeheugen te laten lezen (**laden**).

CLOAD "bestandsnaam"

Het CLOAD bevel laadt een programma met een bepaalde naam van de band in het computergeheugen.

Je voert de naam van het bestand dat je van de band in de computer wilt laden in onder "bestandsnaam" in het bevel. In dit geval zou dat zijn:

```
CLOAD "HART"
```

Druk op de ↵ toets en zet de cassetterecorder in de weergavestand. (Wanneer de afstandbedieningsaansluiting met de computer is verbonden en je de PLAY (weergave) toets indrukt, voordat je op de ↵ toets hebt gedrukt, dan zal de band niet beginnen te lopen. De band gaat pas lopen na indrukken van de ↵ toets.) Wanneer de band het punt bereikt waar begonnen werd met opnemen van het programma, dan verschijnt het

```
Found:HART
```

bericht op het scherm. Nadat het volledige programma in het computergeheugen is geladen, zal op het scherm het volgende te zien zijn:

```
CLOAD "HART"
Found:HART
Ok
```

Als er in plaats van "Ok"

```
Device I/O error
```

verschijnt, dan betekent dat dat het programma niet juist van de band in het computergeheugen is geladen. Verander het volumenivo van de recorder en laad het programma opnieuw met het CLOAD bevel.

Geef, nadat het programma is geladen, het LIST bevel om te kontroleren of het programma zich in het geheugen van de computer bevindt. Voer vervolgens het programma uit met het RUN bevel.

PROGRAMMA'S OPSLAAN OP DISKETTE

- Een diskette formateren—CALL FORMAT
- Het programma op diskette opslaan—SAVE
- Kontroleren of het programma is opgeslagen—FILES
- Een programma vanaf diskette laden—LOAD
- Een programma van diskette wissen—KILL

Dit gedeelte is van toepassing op computers die voorzien zijn van diskette-eenheden. Als jouw computer geen diskette-eenheden heeft, sla dan het voorgaande gedeelte "Programma's opslaan op de band" er op na voor het opslaan van programma's.
Ook wanneer jouw MSX-2 computer geen diskette-eenheden heeft, dan is het toch mogelijk programma's op diskette op te slaan door gebruik te maken van een externe diskette-eenheid voor flexibele schijfjes, zoals de Sony HBD-50. In dit geval is het nodig de diskette-eenheid op de computer aan te sluiten en MSX-BASIC te starten. Bekijk hiervoor de onderstaande paragraaf "Gebruik van een externe diskette-eenheid".

Gebruik van een externe diskette-eenheid

(1) Schakel de computer en de externe diskette-eenheid uit.
(2) Steek het interface cartridge van de diskette-eenheid in de cartridge-gleuf van de computer.
(3) Schakel de diskette-eenheid in. Schakel vervolgens de TV of monitor en de computer in. Na een korte pauze kan het volgende op het scherm verschijnen.

```
Enter date (Y-M-D):
```

(4) Als het bovenstaande bericht op het scherm verschijnt, druk je op de [↵] toets.

Hiermee is de startprocedure voor MSX-Disk BASIC voltooid.

EEN DISKETTE FORMATEREN CALL FORMAT

Bij uitschakelen van de computer wordt het programma waar je zo hard aan gewerkt hebt, uit het geheugen van de computer gewist. Daarom moet het programma eerst op de diskette in de diskette-eenheid van de computer worden opgeslagen. Wanneer je dat gedaan hebt kun je, zelfs na uitschakelen van de computer, bij opnieuw inschakelen het programma vanaf de diskette in het computergeheugen laden.

Maar voordat je een programma op een nieuwe diskette kunt opslaan, moet je de diskette eerst **formateren**.

Bij het formateren van een diskette schrijft de computer aan de hand van een vast aantal regels speciale gegevens op de diskette, die dienen als "wegwijzers" voor de computer, zodat deze meteen weet waar elk bestand zich op de diskette bevindt. Het formateren van een diskette is vergelijkbaar met het trekken van lijnen op een leeg vel, zodat je weet waar je moet schrijven en waar je ruimte tussen de regels moet laten. Je hoeft je niet druk te maken over de gegevens voor het formateren van de diskette: dat doet de computer voor je. De enige regel die jij moet onthouden is dat je **een nieuwe diskette ALTIJD moet formateren alvorens deze te gebruiken**.

Opmerking: bij het formateren van een diskette worden alle gegevens die zich op de diskette bevinden gewist.

Laten we eens een nieuwe diskette formateren. De werkwijze is als volgt:

(1) Voer het CALL FORMAT bevel uit.

Bij indrukken van de [↵] toets verschijnt

```
Drive name ? (A,B)
```

op het scherm.

Dit bericht vraagt of je een diskette in diskette-eenheid A of een diskette in diskette-eenheid B wilt formateren. Wanneer je slechts één diskette-eenheid gebruikt, dan is dit A. In dit geval druk je natuurlijk op de [A] toets.

(2) Als je jouw diskette-eenheid voor dubbelzijdige diskettes kunt gebruiken, verschijnt vervolgens

```
1 - Single sided, 9 sectors
2 - Double sided, 9 sectors
```

op het scherm. Druk op de [1] toets als jouw diskette een enkelzijdige diskette is. Is het een dubbelzijdige diskette, druk dan op de [2] toets.

Bij indrukken van de toets verschijnt

```
Strike a key when ready
```

op het scherm. (Bij gebruik van een diskette-eenheid voor enkelzijdige diskettes, zoals de SONY HBD-50, verschijnt dit bericht bij indrukken van de [A] toets.)

(3) Steek een een nieuwe diskette in de diskette-eenheid en sla een willekeurige toets aan. Hierna formateert de computer de diskette. Wanneer het formateren voltooid is, verschijnt het

```
Format complete
Ok
```

bericht op het scherm.

HET PROGRAMMA OP DISKETTE OPSLAAN SAVE

Het SAVE bevel wordt gebruikt om het programma dat zich in het computergeheugen bevindt in brugtaal op een diskette op te slaan.

SAVE "A: bestandsnaam. soortnaam"

A: geeft aan dat het om de A diskette-eenheid gaat. Bij gebruik van slechts één diskette-eenheid kun je A weglaten.

De **bestandsnaam** is de naam die je aan een bestand geeft om het te kunnen onderscheiden van andere bestanden.

Er zijn vier regels die je moet onthouden bij het geven van een bestandsnaam aan een programma:

- De naam kan uit maximaal 8 letters bestaan
- Je kunt als aanvulling op de letters van het alfabet cijfers en tekens gebruiken
- De computer maakt geen onderscheid tussen kleine en hoofdletters
- De volgende (lees)tekens kunnen niet gebruikt worden voor een bestandsnaam:

  , . ; : " * ? en spatie

De **soortnaam** geeft, zoals de naam al zegt, het soort bestand aan. Je schrijft hiervoor een punt (.) gevolgd door een naam van maximaal drie letters. Laten we het bestand dat we gaan opslaan als soortnaam "BAS" geven, aangezien het een BASIC bestand is. Deze naam geeft aan dat het om een programma gaat dat in BASIC is geschreven. De soortnaam kan weggelaten worden, maar als je meerdere soorten bestanden op je diskette gaat opslaan dan zul je zien dan het handig is om het soort bestand aan te geven in het SAVE bevel. Leer jezelf dit vanaf het begin aan, evenals het gebruik van "BAS" als soortnaam voor alle programma's die je in BASIC hebt geschreven.

Laten we nu een programma de bestandsnaam HART en de soortnaam .BAS geven en dit programma op diskette opslaan.

Geef het volgende bevel:

```
SAVE "HART.BAS"
```

Bij indrukken van ↵ toets zul je de diskette-eenheid horen werken. Wanneer het programma is opgeslagen verschijnt er "Ok" op het scherm.

KONTROLEREN OF HET PROGRAMMA IS OPGESLAGEN FILES

Het FILES bevel wordt gebruikt om te kontroleren of het programma op diskette is opgeslagen.

FILES

Na het geven van het FILES bevel worden alle bestandsnamen en soortnamen van alle bestanden die op diskette zijn opgeslagen op het scherm weergegeven.

```
FILES
HART    .BAS
OK
```

Bij uitvoeren van het FILES bevel moet de naam, HART.BAS, van het programma dat je net hebt opgeslagen, op het scherm verschijnen. Wanneer er naast dit bestand ook andere bestanden op dezelfde diskette zijn opgeslagen, dan verschijnen de namen van deze bestanden ook.

EEN PROGRAMMA VANAF DISKETTE LADEN
LOAD

Maak gebruik van het NEW bevel om het computergeheugen leeg te maken, alvorens een programma vanaf diskette in het geheugen te laden.

Het LOAD bevel wordt gebruikt om een programma vanaf een diskette in het geheugen van de computer te laden.

LOAD "A: bestandsnaam. soortnaam"

A: kan weggelaten worden (zie voor bijzonderheden hoofdstuk 9 in "BASIC VOOR GEVORDERDEN"). De bestandsnaam en de soortnaam zijn dezelfde namen als die je gebruikte bij het opslaan van het programma. Je voert dus:

```
LOAD "HART.BAS"
```

in.
Bij indrukken van de ↵ toets gaat de diskette-eenheid aan het werk en het programma wordt geladen. Wanneer het laden beëindigd is, verschijnt er "Ok" op het scherm. Kontroleer met behulp van het LIST bevel of het programma zich in het computergeheugen bevindt en voer het programma met behulp van het RUN bevel uit.

Iedere keer dat je een bepaald programma, dat op diskette is opgeslagen, wilt gebruiken kun je dat met het LOAD bevel in het geheugen van de computer laden.

EEN PROGRAMMA VAN DE DISKETTE WISSEN
KILL

Naarmate je meer programma's op de diskette opslaat, des te voller wordt de diskette.

KILL is het bevel dat je kunt gebruiken om overbodige bestanden van de diskette te wissen.

KILL "A: bestandsnaam. soortnaam"

A: kan weggelaten worden. Om het HART .BAS programma dat je op de diskette hebt opgeslagen, te wissen, voer je het

```
KILL "HART.BAS"
```

bevel in.

Hoofdstuk 3
Lijstvariabelen

EEN PROGRAMMA MET LIJSTVARIABELEN

- Afmetingen van lijstvariabelen bepalen met DIM

GEBRUIK VAN LIJSTVARIABELEN DIM

Laten we eens wat lijstvariabelen in een werkelijk programma gaan toepassen. We kiezen hiervoor twee lijstvariabelen van het lettertekenrij-type, die we N$(N) en T$(N) noemen. We kiezen de afmetingen zo dat ze beide vijf verschillende gegevens kunnen bevatten, m.a.w. de waarde van N loopt van 0 t/m 4.
We gebruiken de N$(N) lijstvariabele voor een aantal namen van mensen die we kennen, en de T$(N) lijstvariabele voor hun telefoonnummers. De gegevens luiden als volgt:

Lijstvariabele	Gegeven	Lijstvariabele	Gegeven
N$(0)	PETER	T$(0)	111-2222
N$(1)	PAUL	T$(1)	222-3333
N$(2)	MARY	T$(2)	333-4444
N$(3)	TOM	T$(3)	444-5555
N$(4)	SUSAN	T$(4)	555-6666

Gegevens aan lijstvariabelen toewijzen
Net als dat bij gewone variabelen gaat, worden ook aan lijstvariabelen de gegevens als waarden toegewezen met behulp van de bevelen LET, INPUT of READ—DATA. In dit geval gebruiken we het READ—DATA bevel.

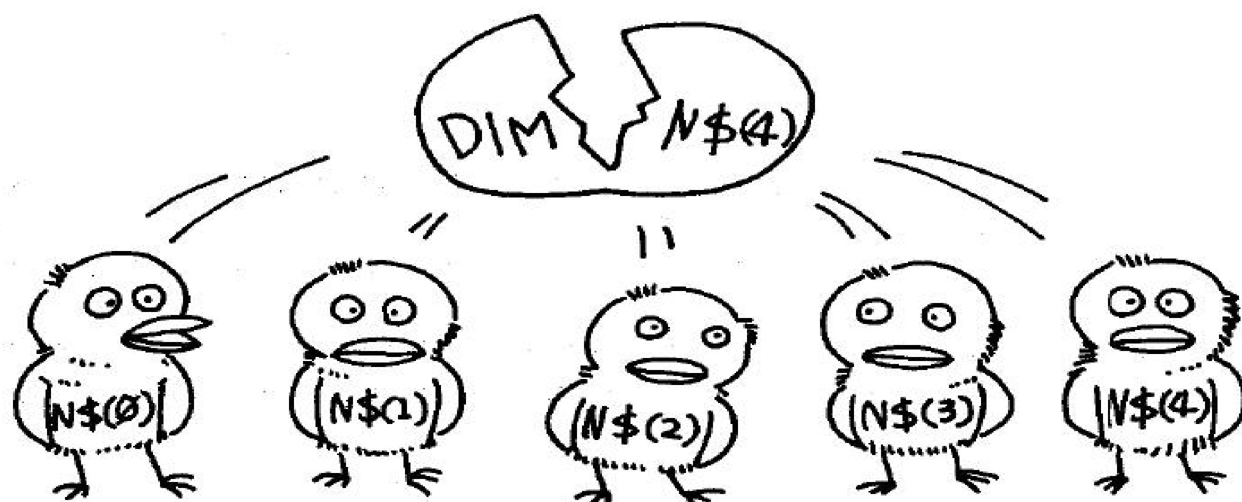
```
10 DIM N$(4),T$(4)
20 FOR L=0 TO 4
30 READ N$(L),T$(L)
40 NEXT L
100 END
110 DATA PETER,111-2222
120 DATA PAUL,222-3333
130 DATA MARY,333-4444
140 DATA TOM,444-5555
150 DATA SUSAN,555-6666
```

In regel 10 hebben we een geheel nieuwe instruktie gebruikt.

```
10 DIM N$(4),T$(4)
```

De DIM instruktie dient om het aantal lijstvariabelen in een programma te bepalen, de afmetingen van elke lijstvariabele, en daarmee het totaal aantal variabelen waarover je kan beschikken.

Zo betekent DIM N$(4), T$(4) dat in dit programma de lijstvariabele N$, die 5 variabelen van N$(0) t/m N$(4) omvat, en de lijstvariabele T$, die 5 variabelen van T$(0) t/m T$(4) omvat, gebruikt zullen worden.

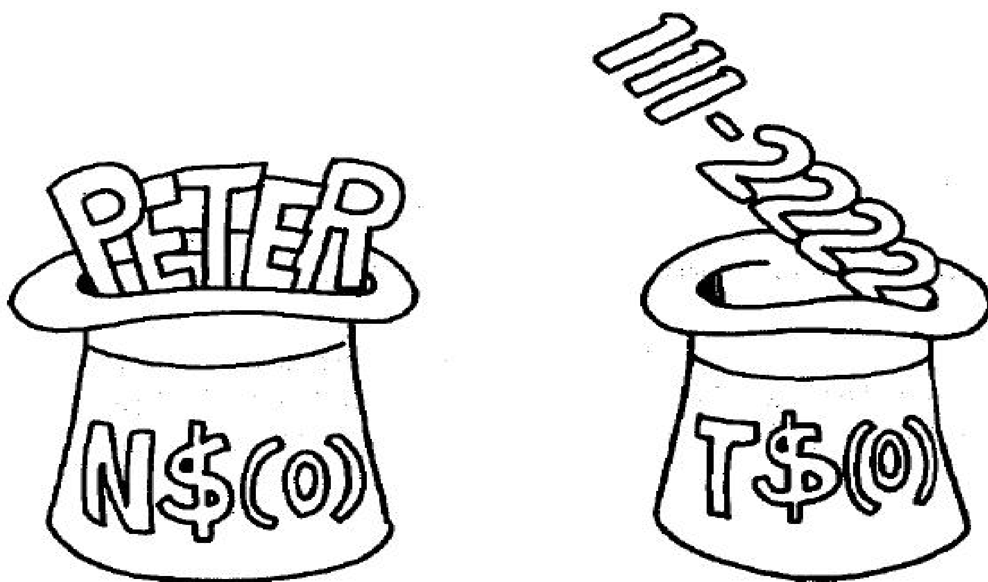


DIM lijstvariabele (maximale waarde)

De DIM instruktie geeft de maximale waarde van het aantal variabelen binnen de ( ) haakjes van een lijstvariabele aan. Als de maximale waarde N is, dan geeft de lijstvariabele je de beschikking over een aantal variabelen genummerd van 0 t/m N, oftewel een aantal van N + 1 variabelen.

Alvorens in een programma een lijstvariabele langer dan 10 elementen gebruikt kan worden, moeten altijd eerst de naam en de afmetingen hiervan met een DIM instruktie vastgelegd worden.

Nadat in regel 10 de lijstvariabelen vastgelegd zijn, wijst de FOR—NEXT lus in de regels 20 t/m 40 gegevens aan de lijstvariabelen toe. De eerste keer dat de lus doorlopen wordt is de waarde van L gelijk aan 0, zodat het READ bevel in regel 30 de waarde PETER aan variabele N$(0) toewijst en de waarde 111-2222 aan T$(0).



Vervolgens worden de andere gegevens op soortgelijke wijze aan N$(1), T$(1) enzovoort toegewezen. Het programma is ten einde nadat de laatste lus SUSAN aan N$(4) en 555-6666 aan T$(4) toegewezen heeft.
Je kan kontroleren of de gegevens op de juiste wijze aan de variabelen zijn toegewezen door rechtstreeks een PRINT bevel uit te doen voeren.

Wanneer je invoert

```
PRINT N$(0),T$(0)
```

dan verschijnt op het scherm

```
PRINT N$(0),T$(0)
PETER           111-2222
Ok
```

Als je PRINT N$(1) invoert, dan komt er PAUL te staan, en met PRINT T$(2) komt er 333-4444 te staan.

Hiermee eindigt de inleiding voor beginners in MSX2-BASIC. Als je de beginselen van BASIC inmiddels onder de knie hebt, dan zijn wij dik tevreden.
Je kunt nu de geleerde BASIC bevelen gebruiken om je eigen programma's te schrijven. De beste manier om vooruitgang te boeken in het schrijven van programma's is om op eigen initiatief veranderingen aan te brengen in de programma's die je met behulp van dit boek hebt gemaakt.
Met een beetje denkwerk zul je al gauw ontdekken dat er vele manieren zijn waarop deze programma's te veranderen zijn, zodat ze gemakkelijker te gebruiken zijn. En wees vooral niet bang om fouten te maken wanneer je begint met het schrijven van je eigen programma's. Zelfs wanneer er sprake is van een vergissing in je programma zal dat de computer niet beschadigen. De computer zal gewoon de vergissing lezen en vervolgens een foutmelding op het beeldscherm laten verschijnen om je te vertellen wat en waar je iets verkeerd hebt gedaan. Mocht er een foutmelding op je scherm verschijnen, maak dan gebruik van het LIST bevel om het programma weer te geven en kijk wat je verkeerd hebt gedaan. Verschillende dingen uitproberen, fouten korrigeren: met vallen en opstaan leer je een goede BASIC programmeur te worden.

Hoofdstuk 4
Het Inschakelgeheugen

DE SET INSTRUKTIE

- Het inschakelgeheugen
- Een titel geven—SET TITLE
- Veranderen van de oproep—SET PROMPT
- Instellen van een wachtwoord—SET PASSWORD
- De plaats van het beeld op het scherm veranderen—SET ADJUST
- Kiezen van de pieptoon—SET BEEP
- Bepalen van de beginstatus van het scherm—SET SCREEN

HET INSCHAKELGEHEUGEN

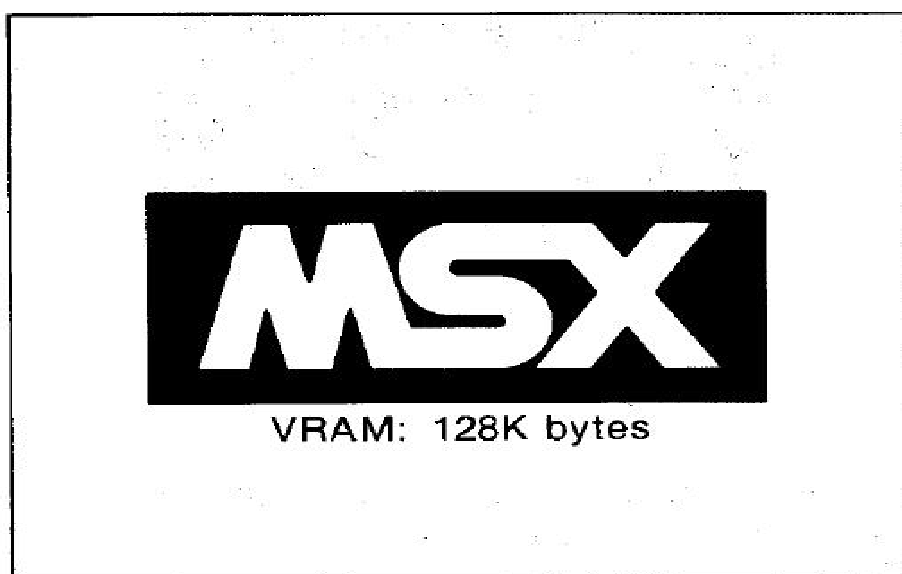
In MSX2-BASIC is er een speciale funktie die je kunt gebruiken om de inschakelinstelling—de instelling bij starten van BASIC—te veranderen. Bovendien kan met behulp van deze funktie de veranderde instelling opgeslagen worden in het inschakelgeheugen (een RAM op batterijvoeding) dat deel uitmaakt van het klokcircuit.

RAM (Random Access Memory) is het hoofdgeheugen van de computer. (In het eerste deel hebben we het hoofdgeheugen vaak "computergeheugen" genoemd.) Wanneer de computer uitgeschakeld wordt, wordt de inhoud van het RAM geheugen gewist. Echter bij een RAM op batterijvoeding blijft de inhoud bewaard, ook wanneer de computer uitgeschakeld wordt. Als we dus in dit speciale RAM geheugen de BASIC inschakelinstelling opslaan, dan wordt bij weer inschakelen van de computer deze instelling gebruikt.

De volgende inschakelinstellingen kunnen veranderd en opgeslagen worden in het inschakelgeheugen (RAM op batterijvoeding).
- Titel en oproep
- Wachtwoord
- De plaats van het beeld op het scherm
- De klank en de sterkte van de pieptoon
- Instellingen op een schermsoort

EEN TITEL GEVEN SET TITLE

Bij het starten van BASIC krijg je, alvorens het "Ok" bericht verschijnt, eerst het volgende op het scherm te zien:



Op de regel onder "VRAM: 128K bytes" (of "VRAM: 64K bytes") op het scherm kun je je eigen titel invullen. De titel wordt bepaald met het SET TITLE bevel.

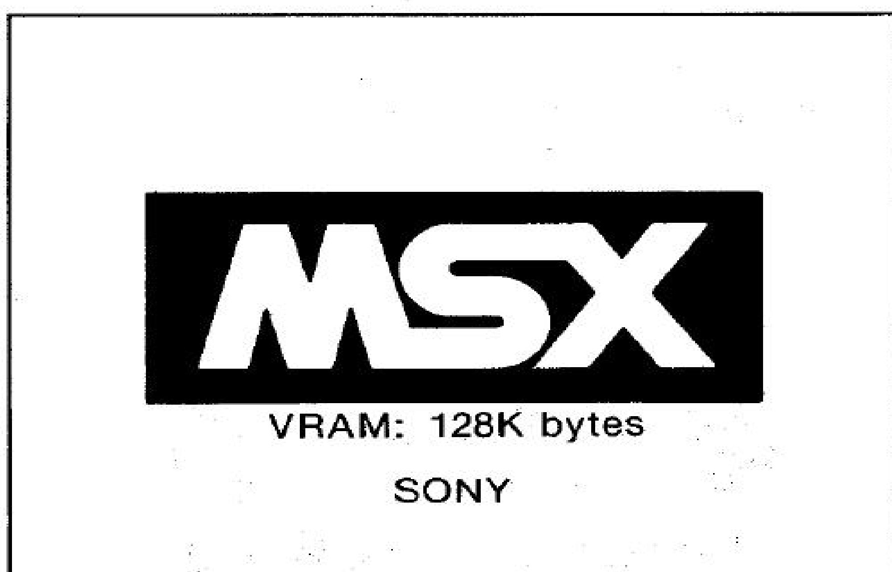
SET TITLE "titel" , [kleur]

De titel kan uit maximaal zes tekens bestaan. **Er kunnen ook vier kleurenkombinaties bepaald worden voor het afbeelden van het MSX** vignet. Als je wilt dan kun je ook de titel weglaten en alleen de kleurenkombinatie bepalen.

Om de titel "SONY" op het scherm te laten verschijnen moet je het volgende losstaande SET TITLE bevel invoeren:

```
SET TITLE "SONY"
```

Druk na het invoeren van dit bevel op de [RESET] toets of schakel de computer uit en weer in. De titel zal nu zoals in de volgende afbeelding getoond wordt, op het scherm te zien zijn.

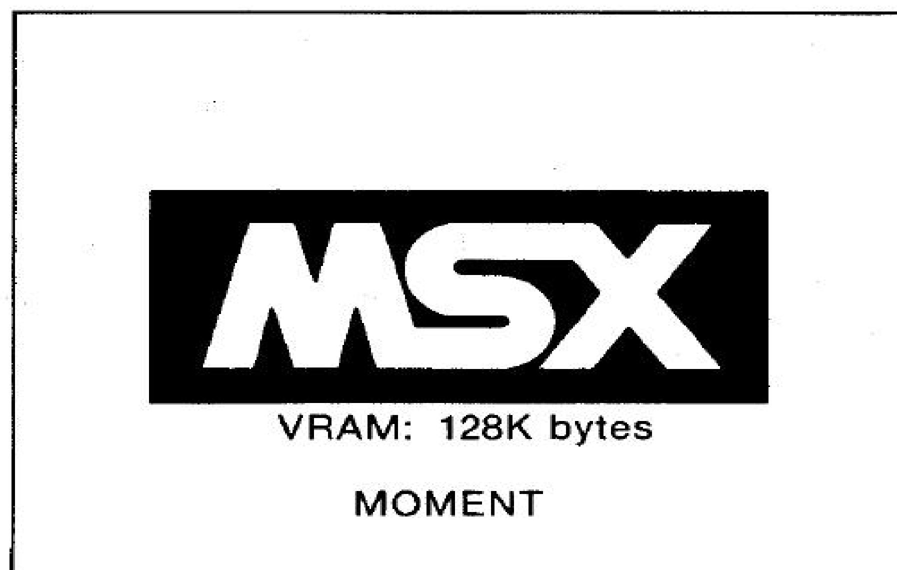


Deze titel zal ook na uitschakelen van de computer in het inschakelgeheugen (RAM op batterijvoeding) bewaard blijven en telkens bij het starten van BASIC te zien zijn.

Bevriezen van de beeldweergave van de titel
Wanneer je een titel kiest die precies uit zes tekens bestaat, dan wordt de beeldweergave bevroren bij het verschijnen van de titel. Als je bijvoorbeeld:

```
SET TITLE "MOMENT"
```

invoert, dan zal de eerstvolgende keer bij het starten van BASIC het volgende te zien zijn:



Omdat "MOMENT" precies uit zes letters bestaat, zal het bovenstaande op het scherm blijven staan totdat er een toets op het toetsenbord wordt aangeslagen. Vervolgens zal "Ok" op het scherm verschijnen en BASIC is gebruiksklaar. Wanneer je in het SET TITLE bevel geen letters, maar zes spaties met behulp van de spatiebalk invult, dan verschijnt er geen titel op het scherm. Alleen het MSX vignet wordt weergegeven, totdat er een toets op het toetsenbord wordt aangeslagen.

Wissen van een titel
Je kunt een titel wissen door

```
SET TITLE ""
```

in te voeren.
Vul geen tekens of spaties tussen de aanhalingstekens in. Bij het invoeren van het SET PROMPT bevel of het SET PASSWORD bevel wordt de instelling op een bepaalde titel eveneens gewist.

Veranderen van de kleur van de weergegeven titel
Het kleurgedeelte in het SET TITLE bevel kan gebruikt worden om op één van de vier kleurenkombinaties van het MSX vignet in te stellen. De kleur bepaal je door een van de nummers 1 t/m 4 in te vullen. Wanneer je

```
SET TITLE "SONY",3
```

invoert, dan geeft het scherm de volgende kleuren te zien:



rood

roodpaars

De onderstaande tabel laat de kleurenkombinaties zien waarop ingesteld kan worden:



Kleur			
1	2	3	4
donker-blauw	donker-groen	rood	donker-geel
zwart	donker-blauw	rood-paars	rood

VERANDEREN VAN DE OPROEP
SET PROMPT

Wanneer BASIC gereed is voor het ontvangen van bevelen verschijnt het "Ok" bericht op het scherm. Ok wordt de oproep genoemd. **Je kunt dit woord in iedere andere oproep van maximaal zes letters veranderen.** Het volgende SET PROMPT bevel verandert "Ok" in "GaDoor"

```
SET PROMPT "GaDoor"
```

Wanneer je de oproep met het SET PROMPT bevel hebt veranderd, dan verschijnt bij starten van BASIC steeds dit woord. Dit woord verandert pas wanneer je een ander PROMPT bevel, een SET TITLE of een SET PASSWORD bevel geeft. Bij het geven van een SET TITLE of een SET PASSWORD bevel, verandert de oproep weer in het originele "Ok".

```
SET PROMPT "GaDoor"
GaDoor
```



INSTELLEN VAN EEN WACHTWOORD
SET PASSWORD

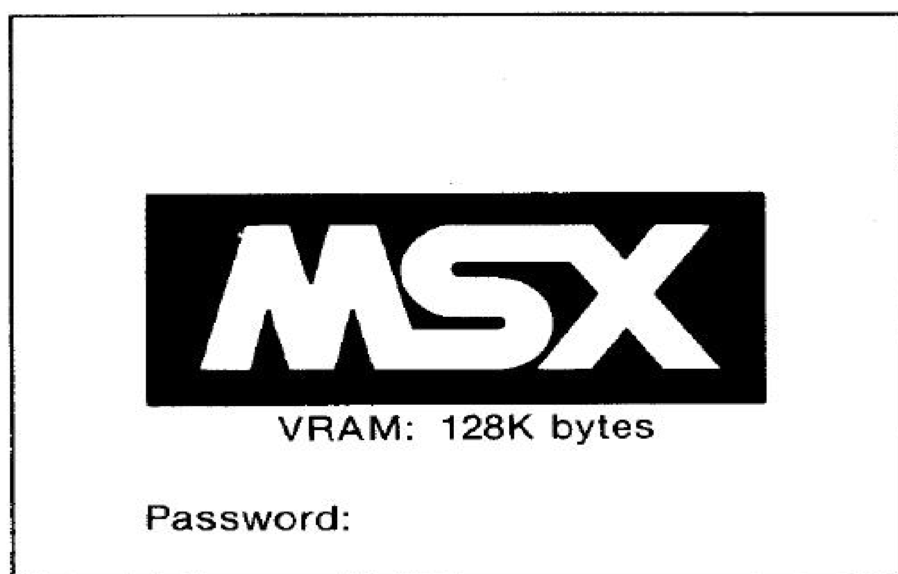
Een wachtwoord is een woord dat alleen jij kent en dat ingevoerd moet worden alvorens BASIC gestart kan worden. Het SET PASSWORD bevel is het bevel waarmee je op een wachtwoord instelt.

SET PASSWORD "wachtwoord"

Het wachtwoord kan uit een lettertekenrij van maximaal 255 tekens bestaan. Als je bijvoorbeeld "IK VIND BASIC LEUK" (18 tekens, inklusief spaties) als wachtwoord wilt nemen, dan voer je het volgende in:

```
SET PASSWORD "IK VIND BASIC LEUK"
```

SET PASSWORD "IK VIND BASIC LEUK"
Wanneer je dit bevel eenmaal hebt uitgevoerd dan zal bij starten van BASIC altijd het volgende te zien te zijn:



totdat het wachtwoord "IK VIND BASIC LEUK" ingevuld is en er op de ↵ toets wordt gedrukt. Wanneer dit wachtwood niet ingevuld wordt of wanneer er een verkeerd wachtwoord wordt ingevuld, dan kan BASIC niet gestart worden. Op deze manier kan alleen jij—of iemand anders die het wachtwoord kent—de computer gebruiken.

Wanneer er eenmaal is ingesteld op een bepaald wachtwoord, dan kan cartridge-programmatuur, zoals spelletjes, niet gebruikt worden totdat het wachtwoord ingevoerd is.

Wissen van het wachtwoord
Het wachtwoord wordt gewist door het invoeren van het SET PROMPT of het SET TITLE bevel. Als bijvoorbeeld

```
SET PROMPT "Ok"
```

ingevoerd wordt, dan verandert de oproep in "Ok" en het wachtwoord wordt gewist.
Samengevat: de laatste SET instruktie——SET PROMPT, SET TITLE of SET PASSWORD—die uitgevoerd moet worden, wordt als de geldende instruktie beschouwd en iedere voorgaande SET instruktie wordt geannuleerd.

Als je het wachtwoord vergeet
Druk de GRAPH, de STOP en de RESET toets tegelijkertijd in wanneer je het wachtwoord vergeten bent. Door deze toetsen ingedrukt te houden, verschijnt het MSX vignet op het scherm. Laat de toetsen los wanneer je gekontroleerd hebt of

```
Password:
```

niet op het scherm verschijnt. BASIC zal vervolgens starten.
Een andere manier om het wachtwoord te omzeilen is door bij inschakelen van de computer de GRAPH en de STOP toets ingedrukt te houden.

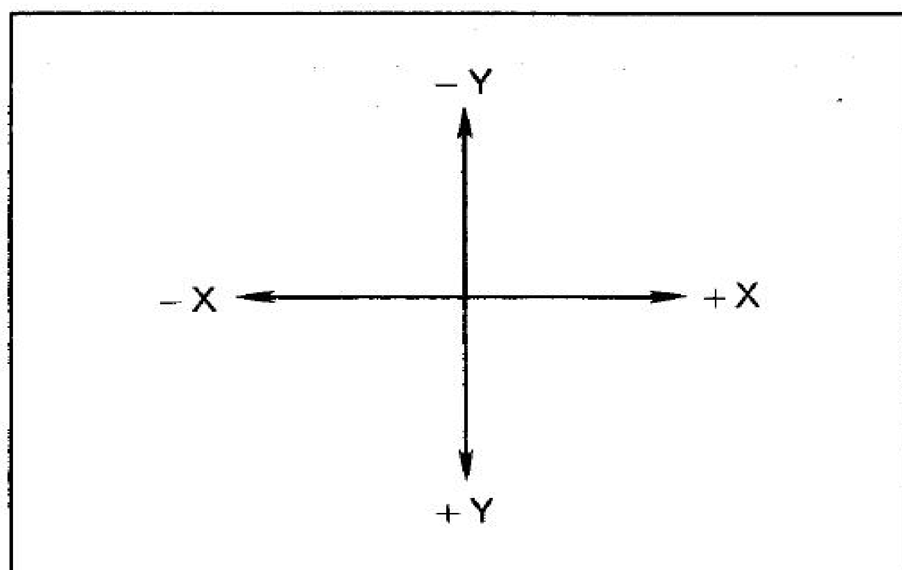
DE PLAATS VAN HET BEELD OP HET SCHERM VERANDEREN SET ADJUST

Het is mogelijk dat het beeld niet altijd precies in het midden van het scherm wordt weergegeven, afhankelijk van de TV of de monitor die je gebruikt. Het SET ADJUST bevel is ervoor om zonodig het beeld naar het midden van het scherm te kunnen verplaatsen.

SET ADJUST (X,Y)

Voor zowel X als Y kun je de cijfers −7 t/m +8 invullen. Bij iedere verandering in de waarde van het ingevulde aantal, verandert de plaats van het beeld op het scherm met één coördinatiepunt.

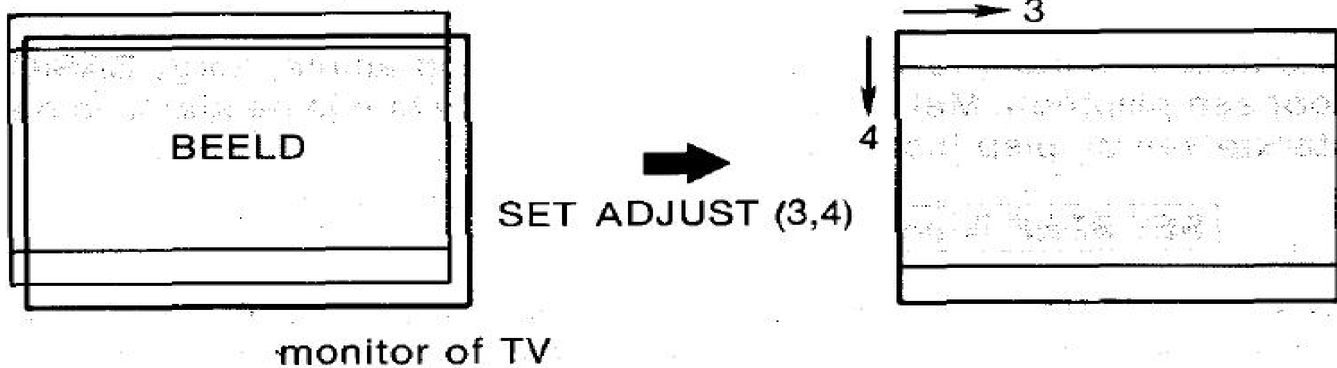
De oorspronkelijke instelling van X en Y is 0. Door een positieve waarde voor X in te vullen wordt het beeld naar rechts verplaatst en door een negatieve waarde voor X in te vullen wordt het beeld naar links verschoven. Een positieve waarde voor de Y verschuift het beeld naar beneden op het scherm en een negatieve waarde naar boven.



Als het beeld zich bijvoorbeeld teveel naar boven en naar links op het scherm bevindt—zoals in de onderstaande afbeelding wordt getoond—dan kun je het centreren door het beeld 3 coördinatiepunten naar rechts en 4 coördinatiepunten naar beneden te verplaatsen. Dit doe je door

```
SET ADJUST (3,4)
```

uit te voeren.



Laten we de SET ADJUST instruktie in een kort programma gebruiken om een interessant effekt te bereiken.

```
10 SCREEN 2
20 CIRCLE (125,95),60
30 FOR Y=-7 TO 8
40 FOR X=-7 TO 8
50 SET ADJUST (X,Y)
60 NEXT X
70 NEXT Y
80 GOTO 30
```

KIEZEN VAN DE PIEPTOON SET BEEP

Wanneer er sprake is van een fout in het programma, zorgt BASIC voor een pieptoon. Met de SET BEEP instruktie kun je de klank en de sterkte van de piep instellen.

SET BEEP [klank] [,volume]

De klank en het volume worden ingesteld door één van de cijfers 1 t/m 4 in te vullen. Wat betreft het volume staat 1 voor het laagste en 4 voor het hoogste volumenivo.

Nummer	1	2	3	4
Klank				
Volume	laagste volume			hoogste volume

Om de klank van de piep op 3 en het volume op 4 in te stellen zou je het volgende moeten invoeren

```
SET BEEP 3,4
```

BEPALEN VAN DE BEGINSTATUS VAN HET SCHERM SET SCREEN

Het SCREEN bevel wordt in hoofdstuk 5 uitgebreid uitgelegd. Hier zullen we alleen een overzicht geven van de instellingen die met het SCREEN bevel kunnen worden gemaakt.

- keuze tekstscherm (SCREEN 0 of 1)
- intoetssignaal ON/OFF
- soort afdrukker
- overdrachtssnelheid cassette
- vervlechtingsfunktie

Met het WIDTH bevel kan het aantal lettertekens per regel ingesteld worden.
De KEY ON/OFF instruktie bepaalt of de funkties van de verschillende funktietoetsen al dan niet onder op het scherm worden weergegeven. Het COLOR bevel wordt gebruikt om de voorgrond-, de achtergrond- en de randkleur te bepalen.

Wanneer SET SCREEN als een losstaand bevel wordt gegeven, dan worden de instellingen die bepaald werden door het SCREEN, het WIDTH, het COLOR bevel en de KEY ON/OFF instruktie als inschakelinstellingen bij starten van BASIC beschouwd.

SET SCREEN

Wanneer bijvoorbeeld

```
SET SCREEN
```

wordt uitgevoerd nadat is ingesteld op de SCREEN 1 schermsoort en de voorgrondkleur op zwart, de achtergrondkleur op grijs en de randkleur op lichtblauw is ingesteld door

```
SCREEN 1
COLOR 1,14,5
```

dan vormen deze kleuren de schermkleuren bij opnieuw starten van BASIC.

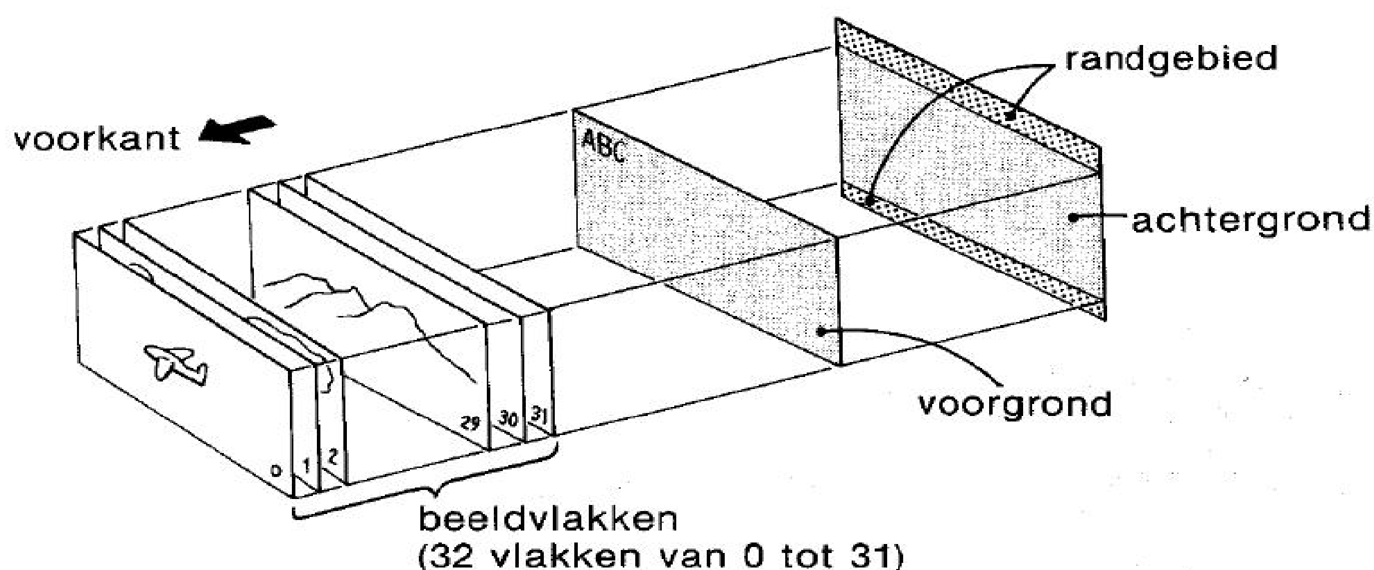
Hoofdstuk 5
Scherm-opbouw en grafische voorstellingen

BEELDSCHERMSOORTEN

- Scherm-opbouw
- Kiezen van de schermsoort—SCREEN
- Het aantal lettertekens per regel instellen—WIDTH
- Grafisch scherm en coördinaten, invullen van STEP

SCHERM-OPBOUW

Het volgende diagram toont de MSX2-BASIC scherm-opbouw



KIEZEN VAN DE SCHERMSOORT SCREEN

Bij het gebruik van het **tekstscherm** worden er karakters (lettertekens, cijfers, e.d.) weergegeven. Bij gebruik van het **grafische scherm** worden er figuren en tekeningen in stippeneenheden weergegeven. Er bestaan in totaal negen tekst- en grafische schermsoorten in MSX2-BASIC. Het SCREEN bevel wordt gebruikt om in te stellen op een bepaalde schermsoort.

SCREEN [schermnummer], [sprite-formaat], [intoetssignaal], [overdrachtssnelheid], [soort printer], [vervlechting]

Het SCREEN bevel bepaalt de schermsoort, het beeldpatroonformaat, het intoetssignaal aan/uit, de overdrachtssnelheid van/naar cassetteband, het soort afdrukker en de vervlechtingsfunktie.

Met het SCREEN bevel kan er uit negen schermsoorten gekozen worden.
(Behalve het gedeelte met betrekking tot de schermsoorten van het SCREEN bevel, worden alle andere gedeelten uitgelegd vanaf bladzijde 147.)

Schermsoorten

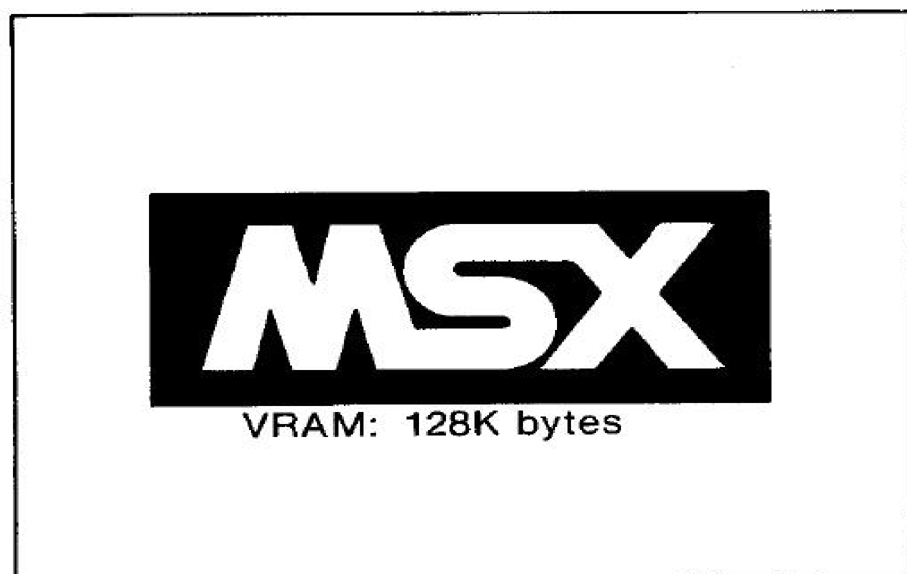
Scherm nummer	Funktie	Detaillering	Kleur	Paginering	Beeldpatronen
0	Tekstscherm	Maximaal 80 lettertekens horizontaal × 24 regels vertikaal	Kleurpaletfunktie, 16 kleuren/512 kleuren	—	niet bruikbaar
1		Maximaal 32 lettertekens horizontaal × 24 regels vertikaal	Kleurpaletfunktie, 16 kleuren/512 kleuren	—	bruikbaar
2	Grafisch scherm, 64K of 128K VRAM geheugen	256 × 192 beeldpunten	Kleurpaletfunktie, 16 kleuren/512 kleuren (2 kleuren/ 8 stippen)	—	bruikbaar
3		64 × 48 beeldpunten, meerkleurig	Kleurpaletfunktie, 16 kleuren/512 kleuren	—	bruikbaar
4		256 × 192 beeldpunten	Kleurpaletfunktie, 16 kleuren/512 kleuren (2 kleuren/8 stippen)	—	bruikbaar (uitgebreide beeldpatroon funktie)
5		256 × 212 beeldpunten	Kleurpaletfunktie, 16 kleuren/512 kleruren	2 pagina's (64K VRAM) 4 pagina's (128K VRAM)	bruikbaar (uitgebreide beeldpatroon funktie)
6		512 × 212 beeldpunten	Kleurpaletfunktie, 4 kleuren/512 kleuren	2 pagina's (64K VRAM) 4 pagina's (128K VRAM)	bruikbaar (uitgebreide beeldpatroon funktie)
7	Grafisch scherm, uitsluitend met 128K VRAM	512 × 212 beeldpunten	Kleurpaletfunktie, 16 kleuren/512 kleuren	2 pagina's	bruikbaar (uitgebreide beeldpatroon funktie)
8		256 × 212 beeldpunten	256 kleuren	2 pagina's	bruikbaar (uitgebreide beeldpatroon funktie)

Bij alle schermsoorten worden alle lettertekens en figuren op het **voorgrondscherm** weergegeven. Achter het voorgrondscherm bevindt zich het **achtergrondscherm**. Je kunt wel de kleur van dit achtergrondscherm veranderen, maar je kunt er geen lettertekens of figuren op weergeven. Er is een **randgebied** aan de boven- en onderkant van het weergavescherm. Net zoals bij het achtergrondscherm kun je wel de kleur van het randgebied veranderen, maar je kunt er niets op weergeven.

Er bevinden zich 32 **beeldvlakken** aan de voorkant van het voorgrondscherm. Deze kunnen bij alle schermsoorten, met uitzondering van de SCREEN 0 schermsoort, gebruikt worden voor het weergeven en bewegen van beeldpatronen. Beeldpatronen en het gebruik hiervan worden in het volgende hoofdstuk uitgelegd.

Opmerking betreffende de afmetingen van het VRAM geheugen

Zoals in het bovenstaande overzicht werd aangegeven, worden de schermsoorten SCREEN 7 en SCREEN 8 alleen gebruikt wanneer het een computer met een 128K bytes VRAM betreft. Bovendien is het aantal pagina's dat gebruikt kan worden bij schermsoort SCREEN 5 en SCREEN 6 afhankelijk de afmetingen van het VRAM.
VRAM staat voor video RAM. Dit is het geheugen waarin de gegevens die weergegeven moeten worden, opgeslagen zijn.
De afmetingen van het VRAM in jouw computer staan vermeld op het MSX vignet dat op het scherm verschijnt bij het starten van BASIC.



De VRAM afmetingen

De weergave bij inschakelen van de computer

TEKSTSCHERM

Om tekst op het scherm weer te geven moet je SCREEN 0 of SCREEN 1 kiezen. Bij de schermsoort SCREEN 0 worden de lettertekens in een 6 stippen (breedte)×8 stippen (hoogte) formaat weergegeven. De breedte van sommige grafische tekens bij een MSX computer is 8 stippen, dus zul je voor het weergegeven van grafische tekens de SCREEN 1 schermsoort moeten gebruiken.

BEPALEN VAN HET AANTAL LETTERTEKENS PER REGEL WIDTH

Bij de SCREEN 0 schermsoort kunnen er in totaal 80 lettertekens per regel weergegeven worden. Bij SCREEN 1 zijn dat er maximaal 32. **Het WIDTH bevel bepaalt bij het tekstscherm het aantal lettertekens dat op iedere regel kan worden weergegeven.**

WIDTH aantal weergegeven lettertekens

Bij SCREEN 0 verschilt de breedte van de lettertekens bij 1 t/m 40 lettertekens per regel van die bij 41 t/m 80 lettertekens per regel.

Wanneer SCREEN 0: WIDTH 40 wordt ingevoerd:

```
ABCDEFG
```

Wanneer SCREEN 0: WIDTH 80 wordt ingevoerd:

```
ABCDEFG
```

Wanneer er ingesteld is op 1—40 lettertekens per regel, dan noemen we dat het **40 tekstscherm**. Is er ingesteld op 41—80 lettertekens per regel dan heet dat het **80 tekstscherm**.

Het aantal lettertekens per regel kun je bij SCREEN 1 instellen op 1 t/m 32. De afmetingen van de lettertekens zijn bij SCREEN 1 altijd hetzelfde.

```
                         ┌80 tekstscherm ┐ ingesteld
             ┌SCREEN 0 ──┤               } met het
Tekstscherm ─┤           └40 tekstscherm ┘ WIDTH bevel
             └SCREEN 1 ── 32 tekstscherm
```

Wanneer je het WIDTH 80 bevel geeft bij het SCREEN 0, 80 tekstscherm, het WIDTH 40 bij het SCREEN 0, 40 tekstscherm en het WIDTH 32 bevel bij het SCREEN 1 tekstscherm, dan zal allereerst het aantal lettertekens het scherm van links naar rechts vullen. Vervolgens zullen bij afname van het aantal lettertekens per regel als gevolg van de verschillende schermsoorten, de weergegeven lettertekens gecentreerd worden.

SCREEN 0: WIDTH 30	SCREEN 1: WIDTH 10
ABCDEFG	ABCDEFG

Het volgende programma laat het soort veranderingen zien dat je in het aantal lettertekens per regel kunt maken.

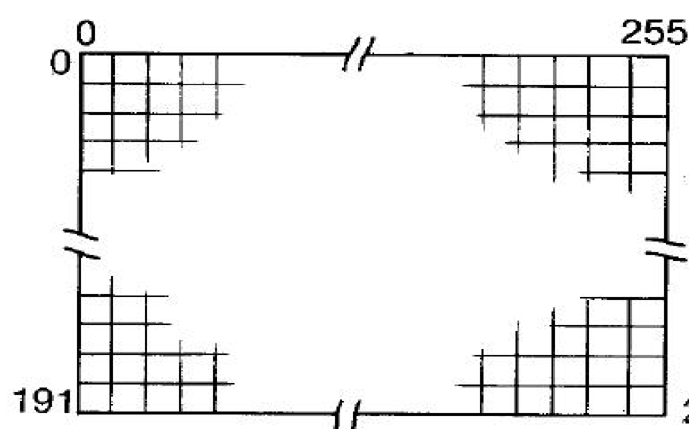
```
10 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 B$="abcdefghijklmnopqrstuvwxyz"
30 SCREEN 0:CLS
40 FOR W=80 TO 10 STEP -10
50 WIDTH W
60 GOSUB 150
70 NEXT W
80 SCREEN 1
90 FOR W=32 TO 12 STEP -10
100 WIDTH W
110 GOSUB 150
120 NEXT W
130 SCREEN 0:WIDTH 40
140 END
150 PRINT "WIDTH";W
160 PRINT:PRINT:PRINT
170 PRINT A$;B$
180 FOR T=0 TO 1000:NEXT T
190 RETURN
```

GRAFISCH SCHERM EN COÖRDINATEN

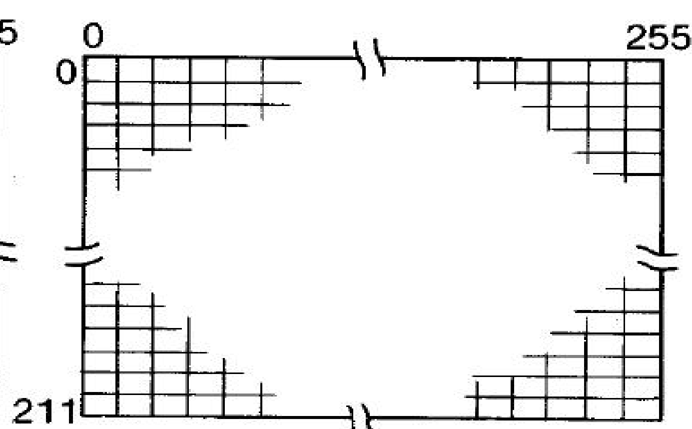
Door in te stellen op SCREEN 2—SCREEN 8 stel je in op één van de grafische schermen. De volgende bevelen worden bij alle grafische schermen gebruikt om tekeningen te maken.

PSET, PRESET ... zet stippen op het scherm
LINE ... tekent lijnen en rechthoeken
CIRCLE ... tekent cirkels, ovalen, bogen en waaiervormen
PAINT ... kleurt een figuur in
DRAW ... tekent figuren die bepaald worden door grafische deelinstrukties

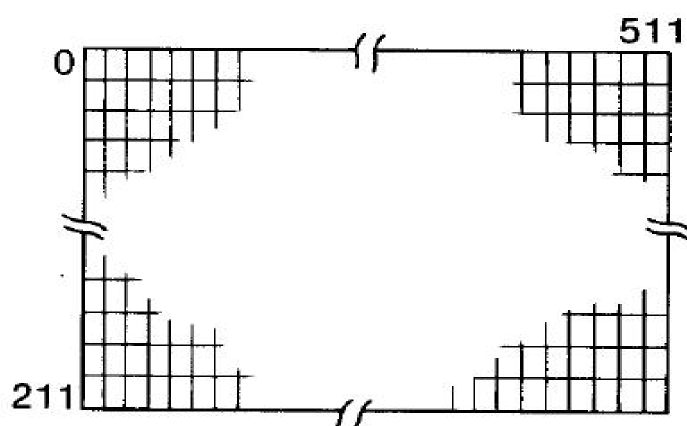
Het beeldscherm is verdeeld in coördinaten die dienen om bij gebruik van de bovenstaande bevelen de plaats op het beelscherm te kunnen bepalen. De coördinaten verschillen per grafische schermsoort.



SCREEN 2, SCREEN 4



SCREEN 5, SCREEN 8



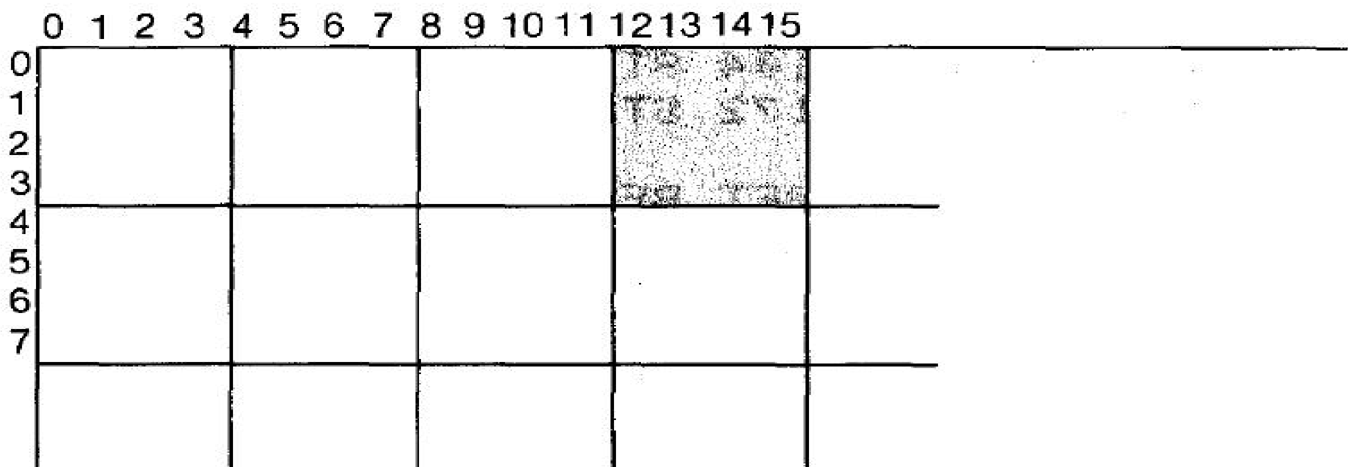
SCREEN 6, SCREEN 7

In het volgende programma worden dezelfde coördinaten in SCREEN 2, SCREEN 5 en SCREEN 6 gebruikt, maar de cirkels die door deze bevelen getrokken worden, bevinden zich telkens op een andere plaats op het beeldscherm.

```
10 SCREEN 2
20 GOSUB 100
30 SCREEN 5
40 GOSUB 100
50 SCREEN 6
60 GOSUB 100
70 END
100 CIRCLE (125,100),90
110 FOR T=0 TO 1000:NEXT T
120 RETURN
```

MEERKLEURENSCHERM (SCREEN 3)

Bij SCREEN 3 worden dezelfde 256 × 192 stipcoördinaten gebruikt als dat het geval is bij SCREEN 2 en SCREEN 4. De eenheid voor het tekenen van een figuur is in dit geval echter een blokje van 4 × 4 stippen.




```
PSET (12,4),1
PSET (14,5),1
PSET (15,7),1
```

De bovenstaande bevelen bijvoorbeeld stellen allemaal in op een plaats binnen hetzelfde 4 × 4 stippenblokje en vandaar dat ieder van deze PSET bevelen het hele blokje zwart kleuren, zoals in bovenstaande afbeelding getoond wordt.

Het LINE bevel

```
LINE (17,5)-(130,110)
```

zal een een ruwe lijn trekken tussen de twee blokken waar de coördinaten (17,5) en (130,110) zich bevinden.

Laten we een programma uitvoeren dat gebruik maakt van het meerkleurenscherm.

```
10 SCREEN 3
20 COLOR ,10,10:CLS
30 LINE (80,76)-(176,108),8,BF
40 FOR Y=80 TO 104 STEP 4
50 FOR X=84 TO 172 STEP 4
60 READ P
70 IF P=0 THEN SET BEEP 1 ELSE 100
80 PSET (X,Y),15:BEEP
90 GOTO 120
100 SET BEEP 2
110 PSET (X,Y),4:BEEP
120 NEXT X
130 NEXT Y
140 GOTO 140
150 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0
160 DATA 0,1,0,1,0,1,1,1,0,1,0,0,0,1,0,0
,0,1,1,1,0,1,0
170 DATA 0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,0
,0,1,0,1,0,1,0
180 DATA 0,1,1,1,0,1,1,0,0,1,0,0,0,1,0,0
,0,1,0,1,0,1,0
190 DATA 0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,0
,0,1,0,1,0,0,0
200 DATA 0,1,0,1,0,1,1,1,0,1,1,1,0,1,1,1
,0,1,1,1,0,1,0
210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0
```

Het SCREEN 3 bevel in regel 10 stelt in op het meerkleurenscherm. In de lus wijst het READ bevel in regel 60 gegevens aan de variabele P toe. Wanneer de waarde van P 0 is, dan zal de stip (het blokje) die door het PSET bevel wordt getekend,in kleur 15 (wit) afgebeeld worden. Wanneer de waarde een ander getal dan 0 is, dan wordt het blokje kleur 4 (donkerblauw). De waarde van P bepaalt eveneens welke van de PSET BEEP bevelen uitgevoerd zal worden en bepaalt daarmee het soort geluid. Hierdoor wordt er bij het uitvoeren van de verschillende PSET bevelen telkens een verschillend geluid geproduceerd.

Bij uitvoering van het programma wordt er het volgende op het scherm getekend:



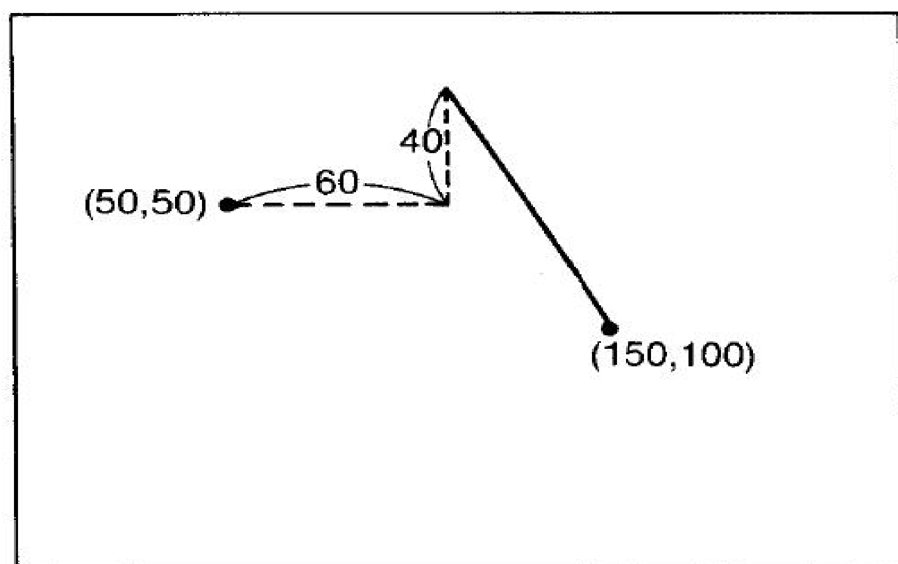
INVULLEN VAN STEP

De STEP (X,Y) deelinstruktie kan bij het PSET, PRESET, LINE, CIRCLE, PAINT en PUT SPRITE bevel gebruikt worden om de (X,Y) coördinaten te bepalen. (PUT SPRITE wordt in het volgende hoofdstuk uitgelegd.)

Wanneer deze grafische bevelen zijn uitgevoerd, wordt het laatst bepaalde punt onthouden. Als daarna de STEP (X,Y) deelinstruktie wordt gegeven, dan wordt de plaats van (X,Y) aan de hand van een nieuw coördinatenstelsel bepaald, waarbij het laatst bepaalde punt als uitgangspunt (0,0) wordt genomen. Bij het weglaten van de STEP (X,Y) deelinstruktie worden de plaatsen aan de hand van het gewone coördinatenstelsel bepaald, waarbij de linker bovenhoek van het scherm als uitgangspunt (0,0) geldt.

```
10 SCREEN 2
20 PSET (50,50)
30 LINE STEP (60,-40)-(150,100)
40 GOTO 40
```

In dit programma wordt het coördinatiepunt (50,50), dat door de PSET deelinstruktie in regel 20 werd ingesteld, onthouden. Vervolgens wordt er in regel 30 gebruik gemaakt van STEP (60, – 40) om het beginpunt voor het LINE bevel te bepalen. (50,50) wordt het nieuwe uitgangspunt en het punt dat zich 60 in de X-richting en –40 in de Y-richting, gerekend vanaf het nieuwe uitgangspunt bevindt, wordt als beginpunt voor de lijn genomen.



Wanneer de STEP deelinstruktie deel uitmaakt van grafische bevelen, wordt de volgende schrijfwijze gebruikt:

PSET STEP(X,Y), kleur
PRESET STEP(X,Y), kleur
LINE STEP(X,Y)—STEP(X,Y), kleur, B / BF
CIRCLE STEP(X,Y), straal, kleur, beginhoek, eindhoek, hoogte/breedteverhouding

Weergeven van lettertekens op een grafisch scherm

Het is ook mogelijk lettertekens op een grafisch scherm weer te geven. Om dit te doen wordt het grafische scherm gebruikt als een zgn. bestandsverwerkend apparaat. Het bestand wordt geopend en de weer de geven lettertekens worden als gegevens naar het bestand gezonden. Zie blz. 254 voor een volledige uitleg.

KLEUREN KIEZEN

- Paletfunktie
- Een palet samenstellen—COLOR
- Kleuren op het SCREEN 8 scherm
- Overlappende kleuren (SCREEN 2 en SCREEN 4)
- Herstellen van de oorspronkelijke kleuren—COLOR

DE KLEURCODE EN DE PALETFUNKTIE

Bij het SCREEN 2 scherm kunnen 16 kleuren gebruikt worden en iedere kleur heeft een code. We geven hieronder een overzicht van de kleurencodes.

Kleurencodetabel

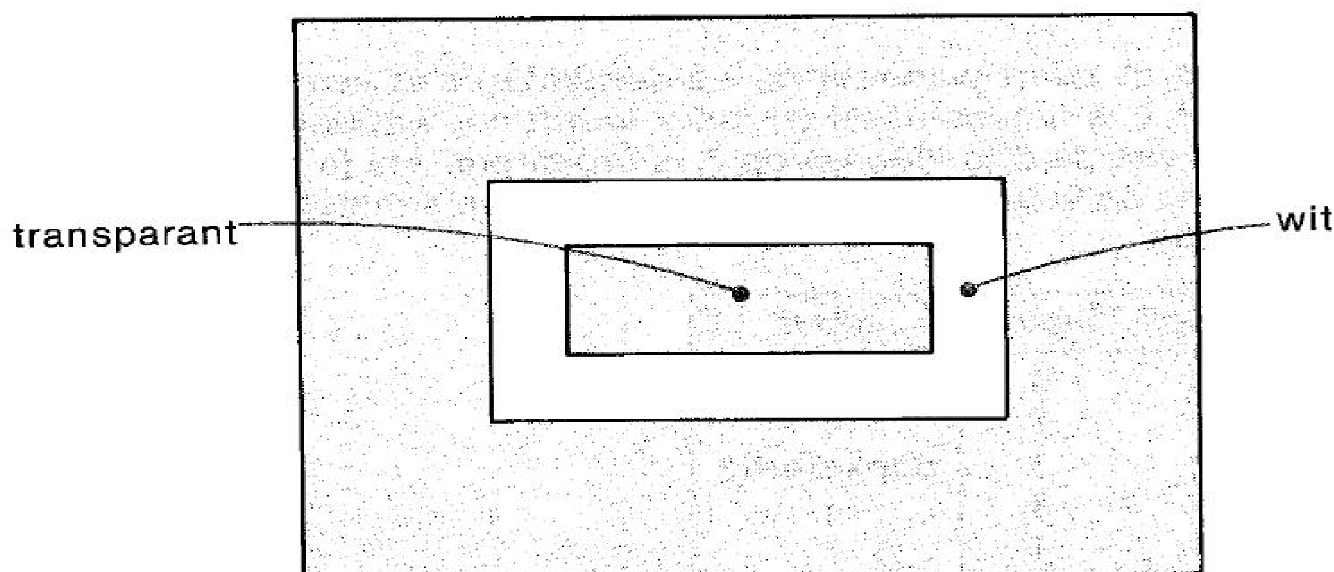
Code	Kleur	Code	Kleur	Code	Kleur	Code	Kleur
0	Transparant	4	Donkerblauw	8	Middelrood	12	Donkergroen
1	Zwart	5	Lichtblauw	9	Licht rood	13	Roodpaars
2	Middelgroen	6	Donkerrood	10	Donkergeel	14	Grijs
3	Lichtgroen	7	Hemelsblauw	11	Lichtgeel	15	Wit

De "kleur" transparant

Wanneer de kleurcode 0 wordt ingesteld, dan is de kleur transparant. Dat wil zeggen dat bij het tekenen van figuren op de voorgrond, de achtergrondkleur bij de figuur doorschijnt. Je kunt dit aan de hand van het volgende programma kontroleren.

```
10 SCREEN 2
20 FOR B=2 TO 14
30 COLOR ,B,B:CLS
40 LINE (50,50)-(180,140),15,BF
50 LINE (70,70)-(160,120),0,BF
60 FOR T=0 TO 1000:NEXT T
70 NEXT B
80 COLOR ,4,4:CLS
90 END
```

De FOR—NEXT lus verandert achtereenvolgens de achtergrondkleur en de randkleur van 2 (middelgroen) naar 14 (grijs). Er wordt een wit vierkant (regel 40) getekend en daar binnenin een kleiner, transparant vierkant (regel 50).
Iedere keer dat de achtergrondkleur verandert, verandert het kleine vierkant in dezelfde kleur, gezien het feit dat de achtergrondkleur door de transparante kleur heen schijnt.



Het kleurenpalet
Zoals op bladzijde 92 getoond werd, zijn de schermsoorten die gebruik maken van de 16 kleuren (kleurcode 0 t/m 15): SCREEN **0**, SCREEN **1**, SCREEN **2**, SCREEN **3**, SCREEN **4**, SCREEN **5** en SCREEN **7**.
De 16 kleuren die in de bovenstaande tabel staan, zijn de 16 kleuren die bij starten van BASIC gebruikt kunnen worden. Maar dit zijn lang niet alle kleuren die mogelijk zijn met de MSX2. De kleurcodes 0 t/m 15 kunnen gebruikt worden om 512 verschillende kleuren te maken, wat je maar wilt.
Het is niet mogelijk al deze kleuren namen te geven. Vandaar dat de hoeveelheid rood, groen en blauw aangegeven wordt—bijvoorbeeld rood 3, groen 2 en blauw 7. Door de hoeveelheid aan te geven kun je een bepaalde kleur samenstellen, net alsof je kleuren op een schilderspalet mengt. Deze funktie wordt de **paletfunktie** genoemd.

Opmerking
Het is ook mogelijk kleuren te bepalen voor kleurcode 0, maar dit is een speciaal geval. In dit gedeelte zullen we uitleggen hoe je de kleurcodes 1 t/m 15 kunt gebruiken om verschillende kleuren samen te stellen.

GEBRUIK VAN DE PALETFUNKTIE

De kleuren rood, groen en blauw kunnen op verschillende helderheidsgraden worden ingesteld: van 0 t/m 7. Deze verschillende kleurinstellingen worden de **helderheid** genoemd. Aangezien er bij ieder van deze drie kleuren acht verschillende gradaties bestaan, kun je door kombinaties te maken van verschillende helderheidsgraden in totaal 512—8×8×8=512—kleuren samenstellen.

De kleur wordt zwart wanneer de helderheidsgraad van rood, groen en blauw op 0 is ingesteld en de kleur wordt wit wanneer de helderheidsgraad van de drie kleuren op 7 is ingesteld. Als je bijvoorbeeld voor alle drie de kleuren de helderheidsgraad op 4 instelt, dan wordt de kleur grijs.

rood	groen	blauw	kleur
0	0	0	zwart
1	1	1	↑
2	2	2	donkergrijs
3	3	3	↑
4	4	4	↓
5	5	5	lichtgrijs
6	6	6	↓
7	7	7	wit

Wanneer je een grotere helderheidsgraad voor één van de drie kleuren instelt (of hoe dichter de helderheidsgraad van de andere twee kleuren in de buurt van de 0 komt), dan zal de kleur met de meeste helderheid overheersen. 7, 0, 0 zorgt bijvoorbeeld voor een puur rode kleur, zoals in de onderstaande voorbeelden van helderheidsinstellingen getoond wordt.

rood	groen	blauw	kleur
4	3	3	grijs, met een licht rood tintje
5	2	2	een kleur die dicht in de buurt van rood komt
5	0	0	rood (een beetje donker)
7	0	0	rood (puur rood, de helderste kleur)
2	0	0	rood dat vrijwel zwart is

EEN PALET SAMENSTELLEN COLOR

Het COLOR bevel wordt gebruikt om te kleurcode en de helderheidsgraad te bepalen die nodig is voor het samenstellen van een bepaalde kleur.

COLOR = (kleurcode, helderheidsgraad rood, helderheidsgraad groen, helderheidsgraad blauw)

Het COLOR bevel wordt gebruikt om de helderheidsgraad van rood, groen en blauw in te stellen van 0 t/m 7 en deze waarden aan een bepaalde kleurcode toe te wijzen.

Om bijvoorbeeld aan kleurcode 5 een helderheidsgraad van 4 voor rood, van 3 voor groen en een helderheidsgraad van 1 voor blauw toe te wijzen, moet je het volgende invoeren:

```
COLOR=(5,4,3,1)
```

```
10 SCREEN 5
20 COLOR=(1,7,7,7)
30 FOR C=2 TO 9
40 COLOR=(C,0,0,C-2)
50 NEXT C
60 COLOR ,1,1:CLS
70 FOR CC=2 TO 9
80 R=100-CC*10
90 CIRCLE (125,100),R,CC
100 PAINT (125,95),CC
110 NEXT CC
120 GOTO 120
```

De kleuren voor de kleurcodes van 1 t/m 9 worden in regel 20 t/m 50 van dit programma bepaald.

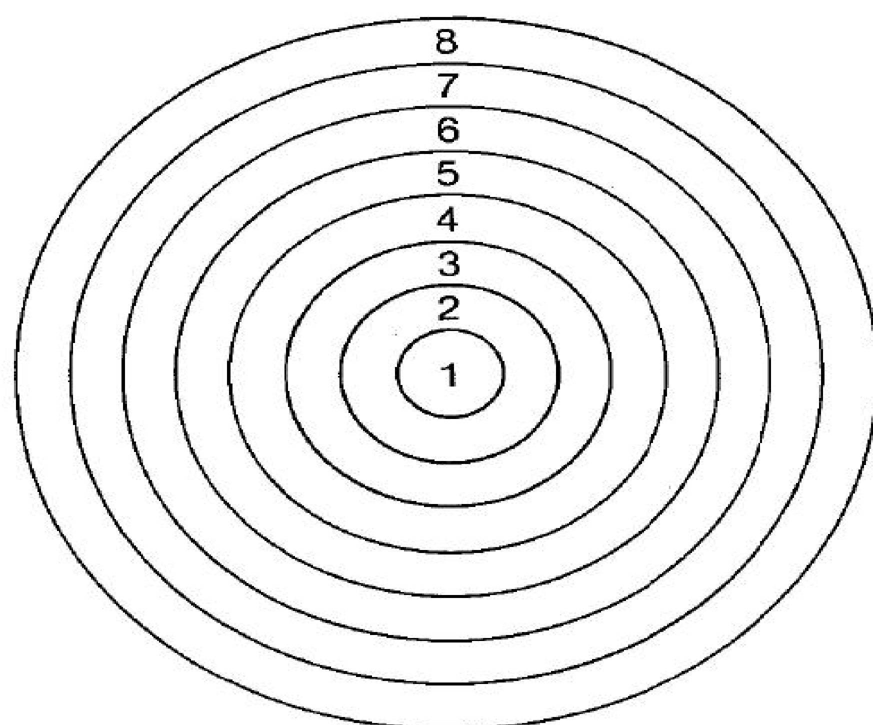
code	rood	groen	blauw
1	7	7	7
2	0	0	0
3	0	0	1
4	0	0	2
5	0	0	3
6	0	0	4
7	0	0	5
8	0	0	6
9	0	0	7

Code 1 is wit, code 2 is zwart en bij de codes 3 t/m 9 verandert de kleur in stapjes van een blauwe kleur die bijna zwart is, naar een puur blauw kleur.
In het COLOR bevel in regel 60 is de achtergrondkleur en de randkleur ingesteld op kleurcode 1, maar in dit geval korrespondeert kleurcode 1 met wit in plaats van met zwart. Vanaf regel 70 worden er cirkels in verschillende tinten blauw in de richting van het middelpunt getrokken, waarbij de cirkel die het laatste getrokken wordt, een puur blauwe cirkel is.
De paletfunktie kan op bovenstaande wijze gebruikt worden om een tekening te maken waarin sprake is van verschillende tinten van een bepaalde kleur.
In het bovenstaande programma wordt gebruik gemaakt van SCREEN 5, wat de beste schermsoort is om snel grafische figuren te tekenen. De tekensnelheid is bij SCREEN 2 en SCREEN 4 lager en er bestaat bovendien meer kans op overlappingen van kleuren. Kleuroverlapping wordt in één van de latere hoofdstukken uitgelegd.

Laten we een ander programma dat gebruik maakt van de paletfunktie schrijven.

```
10 SCREEN 5
20 FOR L=8 TO 1 STEP -1
30 CIRCLE (120,100),L*10+5,L
40 PAINT (120,100),L,L
50 NEXT L
60 K=(K+1) MOD 8
70 FOR L=1 TO 8
80 COLOR=(L,K,K,0)
90 K=(K+1) MOD 8
100 NEXT L
110 GOTO 60
```

De volgende afbeelding toont de kleurcodes voor de concentrische cirkels die met behulp van de FOR-NEXT lus in regel 20 t/m 50 worden getrokken.



In het begin zijn de kleuren die korresponderen met de kleurcodes 1 t/m 8 zwart, middelgroen,. lichtgroen, donkerblauw, lichtblauw, donkerrood, hemelsblauw en middelrood. De cirkels worden in deze kleuren getrokken. Vervolgens wordt de paletfunktie gebruikt om de kleuren die korresponderen met de kleurcodes, te veranderen. Dit gebeurt in regel 60 t/m 110.

In het COLOR bevel in regel 80 wordt gebruik gemaakt van de variabele K om de helderheidsgraad van rood en groen te bepalen. De waarde van K wordt in de regels 60 en 90 ingesteld. De MOD die in deze regels gebruikt wordt, is een van de rekenkundige symbolen, zoals +, −, * en / ook rekenkundige symbolen zijn.
a + b zal a en b optellen, maar **a MOD b geeft de rest bij delen van a door b**. De onderstaande tabel toont het verband tussen K en (K + 1), (K + 1) MOD 8.

K	K+1	(K+1) MOD 8
0	1	1 (1 : 8=0 ... 1)
1	2	2 (2 : 8=0 ... 2)
2	3	3 (3 : 8=0 ... 3)
3	4	4 (4 : 8=0 ... 4)
4	5	5 (5 : 8=0 ... 5)
5	6	6 (6 : 8=0 ... 6)
6	7	7 (7 : 8=0 ... 7)
7	8	0 (8 : 8=1 ... 0)

Wanneer er geen waarde aan een variabele wordt toegekend, dan wordt de waarde van deze variabele op 0 ingesteld. Vandaar dat de waarde van de variabele K voor regel 60 gelijk is aan 0 en na uitvoering van deze regel in 1 verandert. Vervolgens wordt bij uitvoering van K = (K + 1) MOD 8 in regel 90 en 60 de waarde van K telkens met 1 verhoogd. Wanneer K de waarde van 7 heeft bereikt, wordt de waarde de eerstvolgende keer 0 en loopt daarna weer op tot 7.
Als resultaat van de verandering in waarde van K en het COLOR bevel binnen de FOR—NEXT lus, worden de kleuren die corresponderen met de kleurcodes 1 t/m 8 veranderd, als gevolg van de wijziging in helderheidsgraad van rood en groen van 0 t/m 7. Gedurende het hele programma blijft de helderheidsgraad van blauw 0.

Als regel 80 als volgt zou worden veranderd:

```
80 COLOR=(L,0,K,K)
```

dan zou de helderheidsgraad van rood 0 blijven en die van groen en blauw veranderen. Je kunt ook de helderheidsgraad van groen blijvend op 0 instellen en de helderheidsgraad van rood en blauw laten veranderen. Probeer maar eens verschillende kombinaties uit en bekijk het resultaat op het beeldscherm.

HET SCREEN 6 SCHERM EN DE PALETFUNKTIE

Het is ook mogelijk de paletfunktie bij SCREEN 6 te gebruiken. In dit geval kun je echter alleen de kleurcodes van 0 t/m 3, oftewel slechts vier kleuren van de in totaal 512 kleuren, gebruiken. Bij starten van BASIC zijn de kleuren van SCREEN 6 als volgt ingesteld:

code	kleur
0	transparant
1	zwart
2	groen
3	lichtgroen

KLEUREN OP HET SCREEN 8 SCHERM

De paletfunktie wordt bij het SCREEN 8 scherm niet gebruikt, maar het is wel mogelijk met behulp van de kleurcodes 256 kleuren weer te geven.
Bij het SCREEN 8 scherm hebben rood en groen ieder acht verschillende helderheidsgraden, van 0 t/m 7. Er zijn in dit geval vier helderheidsgraden voor blauw, van 0 t/m 3. Gezien het feit dat $8 \times 8 \times 4 = 256$ is, kunnen de kleurcodes van 0 t/m 255 gebruikt worden. Een kleurcode wordt met behulp van de volgende formule bepaald:

kleurcode = 32 × (helderheidsgraad groen) +
4 × (helderheidsgraad rood) +
(helderheidsgraad blauw)

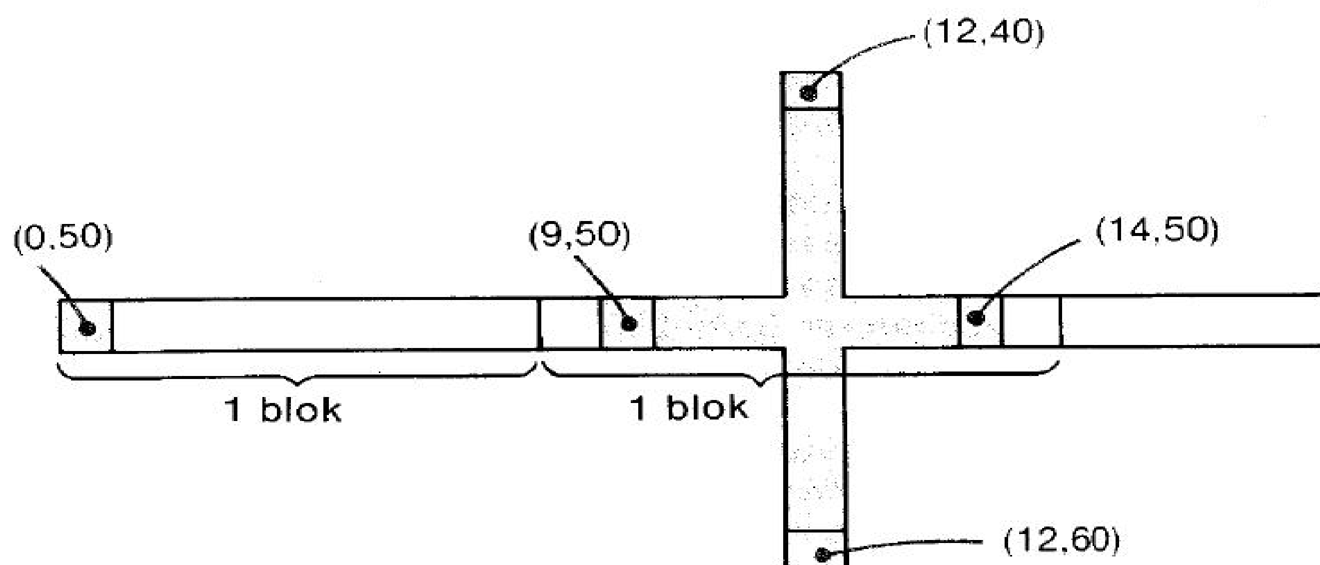
Als de helderheidsgraad van groen bijvoorbeeld 1 is, die van rood 5 en die van blauw 3, dan wordt de kleurcode 55:

$32 \times 1 + 4 \times 5 + 3 = 55$

OVERLAPPENDE KLEUREN OP SCREEN 2 EN SCREEN 4

Bij het SCREEN 2 en het SCREEN 4 scherm kunnen slechts twee kleuren (de achtergrondkleur meegerekend) ingesteld worden voor een blok van 8 horizontale stippen. Wanneer er op meer dan twee kleuren wordt ingesteld, dan wordt de kleur die als laatste bepaald werd, als geldende kleur beschouwd.

```
10 SCREEN 2
20 LINE (9,50)-(14,50),15
30 LINE (12,40)-(12,60),1
40 GOTO 40
```



In dit programma wordt er een horizontale lijn van X9 naar X14 getrokken in het horizontale blok van 8 stippen, dat zich van X8 tot X15 uitstrekt. Vervolgens wordt er op X12 van Y40 naar Y60 een vertikale lijn getrokken die de horizontale lijn snijdt. Dit voegt een derde kleur aan het horizontale blok dat van X8 naar X15 loopt, toe. Ondanks het feit dat er op de kleur wit is ingesteld, zal de horizontale lijn zwart zijn, omdat zwart de laatste kleur is, waarop werd ingesteld. En de laatste kleur waarop werd ingesteld, wordt als geldende kleur voor dit blok beschouwd.

Het op deze wijze "omslaan" van de gekozen kleur in een andere wordt het overlopen van kleur genoemd. Bij het instellen van kleuren voor het SCREEN 2 of het SCREEN 4 scherm is voorzichtigheid geboden om dit overlopen van de kleuren te voorkomen.

Als we regel 20 als volgt wijzigen:

```
LINE (8,50)-(15,50)
```

dan zal de horizontale lijn wel het hele blok van acht stippen van X8 t/m X15 opvullen, zodat de gekozen kleur (in dit geval wit) wel geldig blijft.
Als echter daarna weer een andere kleur voor hetzelfde blok wordt ingesteld, met een bevel als

```
PSET (8,50),8
```

dan verandert de kleur van het gehele blok in de laatst gekozen kleur (in dit PSET bevel dus kleurcode 8).

Onthoud goed dat bij het SCREEN 2 en het SCREEN 4 scherm voor elk blok van 8 stippen slechts maximaal twee kleuren gebruikt kunnen worden.

Bij de meeste andere schermen, SCREEN 5 t/m 8, kunnen de kleuren vrij gekozen worden, in eenheden van 1 stip elk.

HERSTELLEN VAN DE OORSPRONKELIJKE KLEUREN COLOR

Voor het wijzigen van de kleurcodes en kleurinstellingen is het COLOR bevel gebruikt. Om alle kleurinstellingen binnen een programma te doen terugkeren naar de oorspronkelijke instellingen, die gelden bij het starten van BASIC, moet het bevel

COLOR = NEW

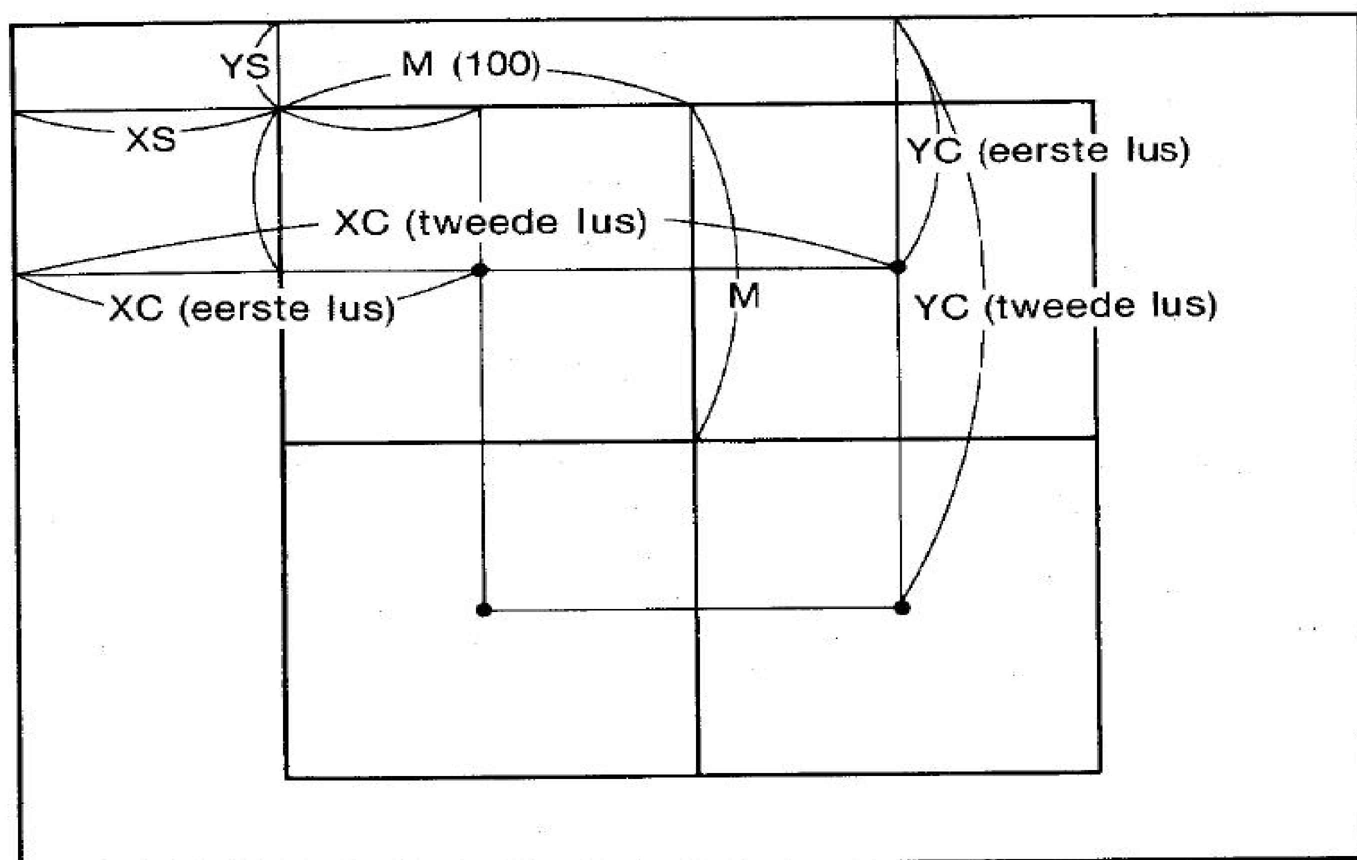
gegeven worden.

Programmeervoorbeeld

Het volgende programma maakt gebruik van de paletfunktie met het SCREEN 5 scherm.

```
10 S=2:CN=10:L=50:M=L*2
20 XS=(255 MOD M)/2:YS=(211 MOD M)/2
30 COLOR 15,0,0:SCREEN 5
40 '-
50 FOR T=0 TO TIME-INT(TIME/100)*100:J=R
ND(1):NEXT
60 ' *** blok tekenen ***
70 FOR XC=L+XS TO 255-L STEP M
80  FOR YC=L+YS TO 211-L STEP M
90  C=0
100    FOR P=L TO 0 STEP -S
110     LINE(XC-P,YC-P)-STEP(P*2,P*2),C+1
,BF
120     C=(C+1)MOD CN
130    NEXT P
140   NEXT YC
150 NEXT XC
160 ' *** kleur bepalen ***
170 R=RND(1)*5+2:G=RND(1)*5+2:B=RND(1)*5
+2
180 FOR P=1 TO CN
190  J=P/CN:R(P)=R*J:G(P)=G*J:B(P)=B*J
200 NEXT P
210 ' *** kleur wijzigen ***
220 FOR K=0 TO 20
230  FOR P=1 TO CN
240   COLOR=(J+1,R(P),G(P),B(P))
250   J=(J+1) MOD CN
260  NEXT P
270  J=(J+1) MOD CN
280 NEXT K
290 GOTO 170
```

In dit voorbeeld komen diverse bevelen voor die nog niet zijn uitgelegd, maar voer deze toch maar letterlijk in zoals ze geschreven zijn, en verwerk dan dit programma.
Hieronder volgt een korte beschrijving van het programma. Allereerst worden door de regels 70 t/m 150 een aantal vierkanten getekend. De variabelen die hierbij gebruikt worden zijn in regels 10 en 20 vastgesteld. De volgende afbeelding toont hoe deze variabelen toegepast worden.



In regels 170 t/m 200 verschijnen de lijstvariabelen R(P), G(P) en B(P). Een aantal waarden van 2 t/m 7 worden toegewezen aan R(1)—R(10), G(1)—G(10) en B(1)—B(10). De RND funktie (voor willekeurige getallen) in regel 170 bepaalt de feitelijk toegewezen waarden. De RND funktie geeft een willekeurig positief (gebroken) getal, groter dan 0 en kleiner dan 1. Zie voor een volledige uitleg van funkties hoofdstuk 7. In de regels 220 t/m 280 worden de waarden van R(P), G(P) en B(P) gebruikt om de kleuren behorend bij kleurcodes 1 t/m 10 te wijzigen met behulp van de kleurpaletfunktie.

Het enkele aanhalingsteken (ofwel apostrof, ') aan het begin van regels 40, 60, 160 en 210 vervangt het woord REM. **De REM (of ') instruktie dient voor het schrijven in een programma van een opmerking die bij het uitvoeren van het programma genegeerd wordt en slechts ter verduidelijking aanwezig is.**

PAGINA'S INSTELLEN

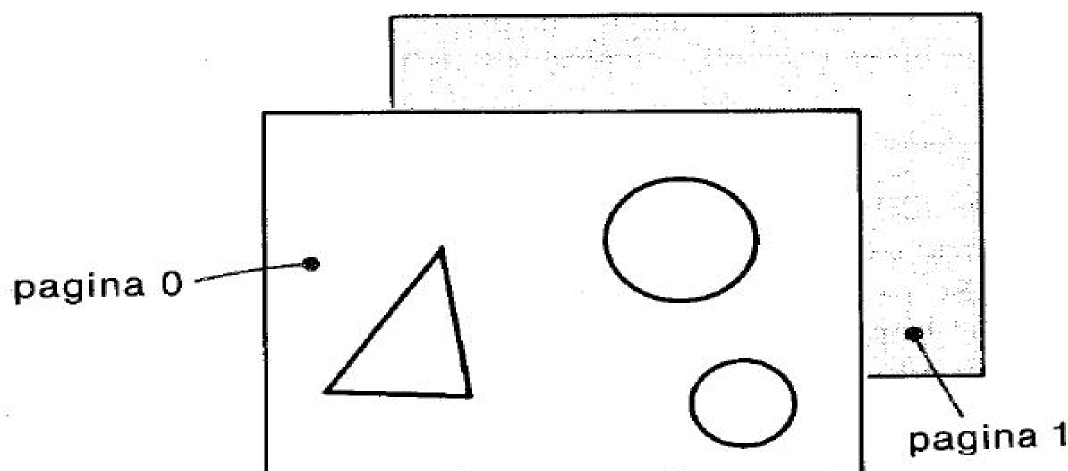
- Omtrent paginering
- De weergegeven pagina en de aktieve pagina
- Pagina's instellen—SET PAGE

GRAFISCHE SCHERMEN EN PAGINERING

Kijk nog eens naar de tabel met SCREEN beeldschermsoorten op blz. 92. Eén van de kolommen is getiteld "paginering". Dit deel van de tabel kan nog iets duidelijker zo geschreven worden:

Beeldschermsoort	VRAM 64K	VRAM 128K
SCREEN 5	2 pagina's	4 pagina's
SCREEN 6	2 pagina's	4 pagina's
SCREEN 7	—	2 pagina's
SCREEN 8	—	2 pagina's

Zoals de naam al aangeeft, is "pagina" vergelijkbaar met een bladzijde van een schrijfblok. Bijvoorbeeld bij 64K VRAM computers kunnen er twee pagina's gebruikt worden voor de SCREEN 5 en de SCREEN 6 schermsoort. Bij het maken van een tekening op het scherm, met behulp van grafische bevelen, wordt slechts één van de twee pagina's gebruikt. De andere pagina blijft leeg.



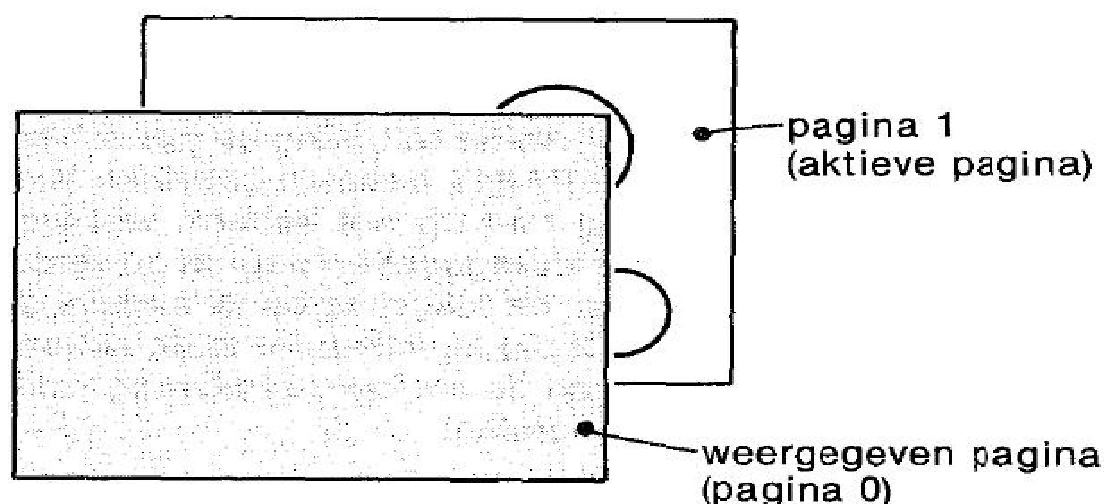
Voor 64K VRAM, SCREEN 5, SCREEN 6

Wanneer je twee pagina's gebruikt, dan krijgen deze **paginanummers**, zoals in de bovenstaande afbeelding getoond wordt. De ene pagina heet pagina 0 en de andere pagina 1. Met een 128K VRAM kunnen er bij SCREEN 5 en SCREEN 6 vier pagina's gebruikt worden. In dit geval zijn de pagina's als 0, 1, 2 en 3 genummerd.

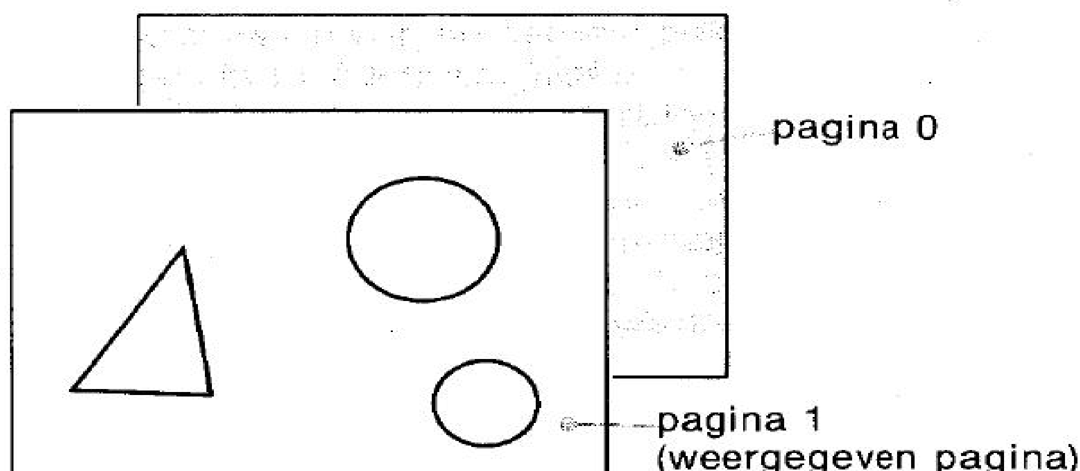
De weergegeven pagina en de aktieve pagina
Wanneer er gebruik kan worden gemaakt van twee of vier pagina's, dan is bij het starten van BASIC pagina 0 altijd de pagina die weergegeven wordt op het beeldscherm van de TV of monitor bij het starten van BASIC. Het is bij starten van BASIC eveneens altijd de pagina waarop tekeningen kunnen worden gemaakt.
De pagina waarop tekeningen kunnen worden gemaakt, wordt de **aktieve** pagina genoemd. De pagina die je op het beeldscherm van je TV of monitor ziet, is de **weergegeven pagina**.

TOEPASSINGEN VAN PAGINERING

Tenzij er een andere instelling is gemaakt, vormt pagina 0 zowel de aktieve als de weergegeven pagina. Daarom wordt bij uitvoering van bevelen zoals CIRCLE en LINE de tekening op pagina 0 gemaakt en wordt deze pagina ook op het beeldscherm weergegeven. Wanneer je echter een tekening maakt terwijl er ingesteld is op pagina 1 als aktieve pagina en pagina 0 als weergegeven pagina, dan wordt de tekening op pagina 1 gemaakt. Gezien het feit dat de pagina die op het scherm wordt weergegeven pagina 0 is, wordt de tekening niet op het beeldscherm weergegeven.



Wanneer je de weergegeven pagina ook op pagina 1 instelt, dan wordt de tekening ook op het beeldscherm weergegeven.



Wanneer in dit geval op pagina 0 als aktieve pagina en op pagina 1 als weergegeven pagina wordt ingesteld, dan worden de volgende tekeningen op pagina 0 gemaakt. Wordt er op pagina 1 als aktieve pagina ingesteld, dan worden de nieuwe tekeningen eveneens op pagina 1 gemaakt.

Gebruik van pagina's

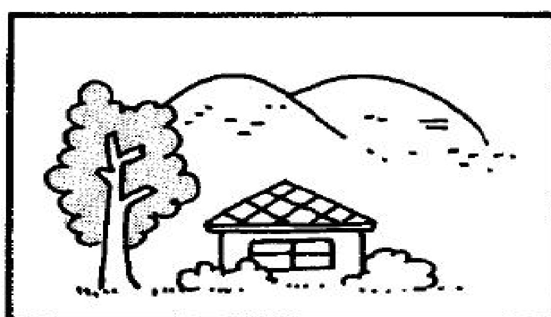
Er zijn vele manieren waarop je de instelmogelijkheid van pagina's kunt gebruiken. Hieronder worden twee manieren beschreven om interessante effekten te verkrijgen.

- **Alleen de voltooide tekening op het scherm weergeven:**
  Het kost tijd om een ingewikkelde tekening op het scherm te maken, zeker wanneer je veel PAINT bevelen gebruikt. Wanneer je het maken van de tekening niet op het scherm wilt weergeven, dan kun je de aktieve en de weergegeven pagina op verschillende paginanummers instellen en de tekening op de aktieve pagina tekenen. Als de tekening voltooid is, verander je de aktieve pagina in de weergegeven pagina en de voltooide tekening wordt in een keer op het scherm weergegeven.



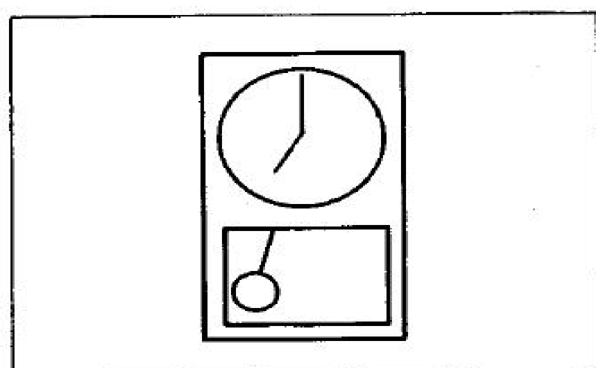
Laat niet zien wat er achter de coulissen gebeurt!

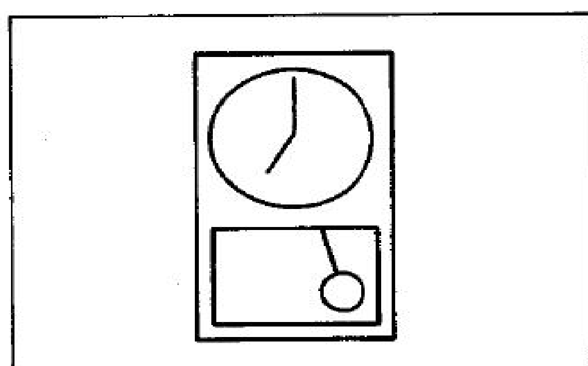
De voltooide tekening verschijnt in één keer

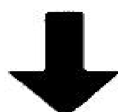


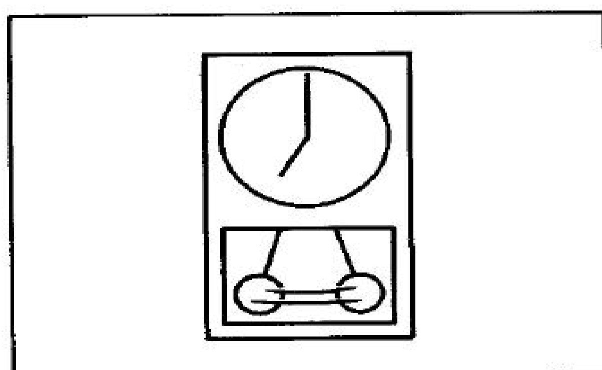
- **Op twee pagina's verschillende tekeningen maken en vervolgens tussen de pagina's heen en weer gaan, zodat er een bewegend effekt wordt verkregen:**

  Je kunt bijvoorbeeld het volgende effekt bereiken:


pagina 0


pagina 1




Ga heen en weer met de weergegeven pagina tussen pagina 0 en pagina 1

INSTELLEN VAN PAGINA'S SET PAGE

Met het SET PAGE bevel kun je de weergegeven en de aktieve pagina instellen op een bepaald paginanummer.

SET PAGE [weergegeven pagina], [aktieve pagina]

Om bijvoorbeeld in te stellen op pagina 0 als de weergegeven pagina en op pagina 1 als aktieve pagina, moet je het volgende uitvoeren:

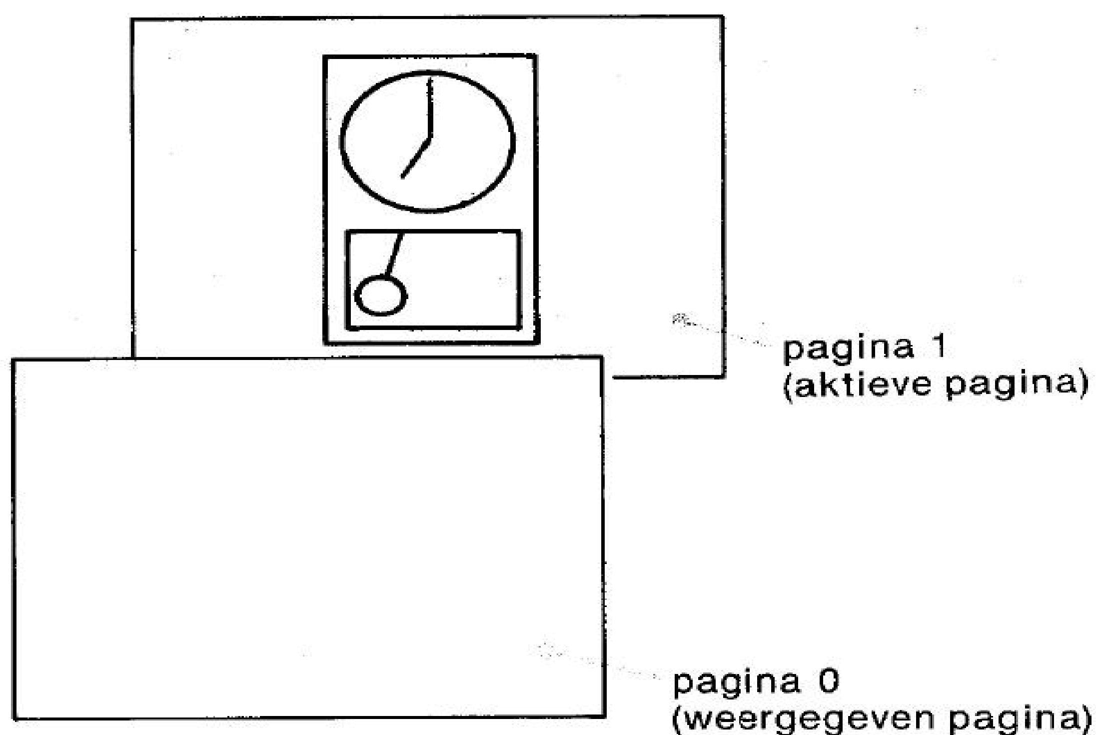
```
SET PAGE 0,1
```

Nadat dit SET PAGE bevel is uitgevoerd, is pagina 0 de pagina die te zien is en pagina 1 de pagina waarop de tekeningen gemaakt worden.

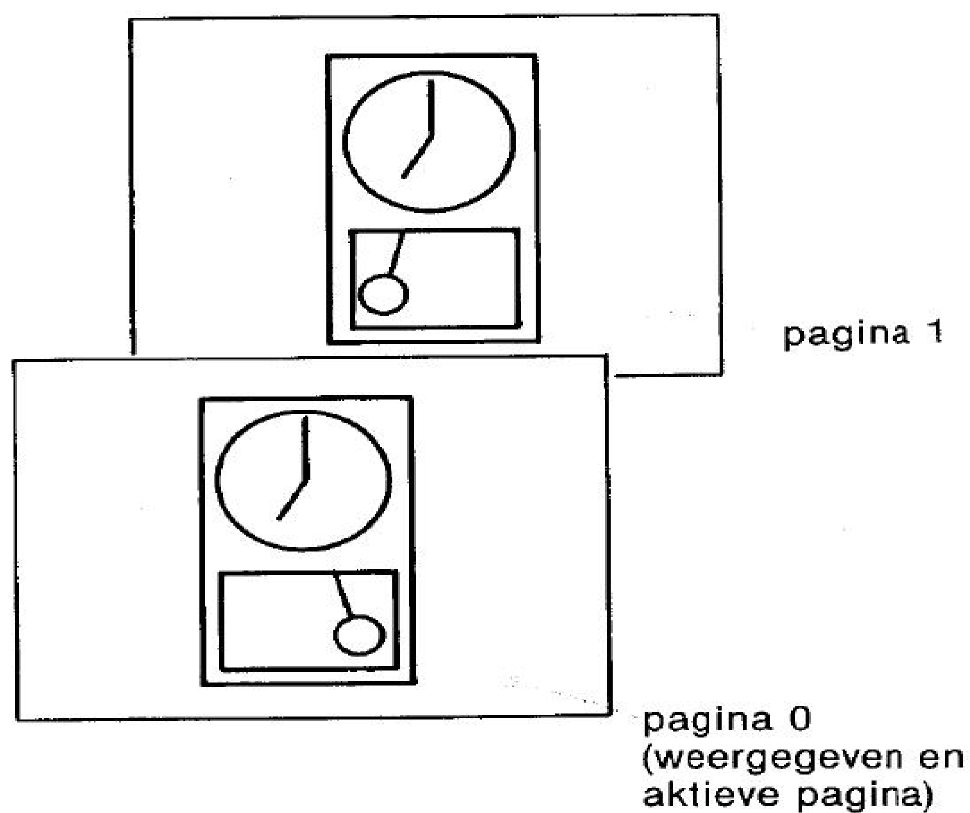
Wisselen van pagina's
Met het volgende programma worden de pagina's telkens verwisseld.

```
10 SCREEN 5
20 SET PAGE 0,1:CLS
30 X1=110:X2=100
40 GOSUB 130
50 SET PAGE 0,0:CLS
60 X1=140:X2=150
70 GOSUB 130
80 SET PAGE 1
90 FOR T=0 TO 300:NEXT T
100 SET PAGE 0
110 FOR T=0 TO 300:NEXT T
120 GOTO 80
130 COLOR 13,3,3:CLS
140 LINE (75,10)-(175,200),,B
150 PAINT (76,11)
160 CIRCLE (125,60),40,15
170 PAINT (125,60),15
180 LINE (125,60)-(125,25),1
190 LINE (125,60)-(115,80),1
200 LINE (80,120)-(170,190),4,BF
210 LINE (X1,120)-(X2,170),10
220 CIRCLE (X2,170),12,10
230 PAINT (X2,171),10
240 RETURN
```

Met dit programma worden er bij de SCREEN 5 schermsoort verschillende tekeningen op pagina 1 en 0 gemaakt.
Allereerst wordt er ingesteld op pagina 0 als de weergegeven pagina en op pagina 1 als aktieve pagina (regel 20).
Vervolgens maakt de subroutine in regel 130 de volgende tekening.



Vervolgens worden de aktieve en de weergegeven pagina beiden ingesteld op pagina 0 (regel 50) en wordt de volgende tekening gemaakt.



De tekening op pagina 1 blijft zoals die is. De lus van regel 80 t/m regel 120 wisselt de weergegeven pagina telkens tussen pagina 1 en 0, hetgeen resulteert in het om de beurt weergegeven van pagina 1 en pagina 0 op het beeldscherm. Dit geeft de indruk alsof de slinger van de klok heen en weer beweegt.

In het SET PAGE bevel in de regels 80 en 100 wordt de aktieve pagina niet ingesteld. Wanneer de aktieve pagina niet ingesteld wordt, dan blijft de vorige instelling op de aktieve pagina van kracht.

GRAFISCHE GEGEVENS KOPIËREN

- Tekeningen kopiëren—COPY
- Kopie van scherm naar scherm
- Kopie van intern geheugen naar scherm en vice versa
- Kopie van scherm naar diskette en vice versa
- Kopie van intern geheugen naar diskette en vice versa
- Logische bewerkingen

TEKENINGEN KOPIËREN

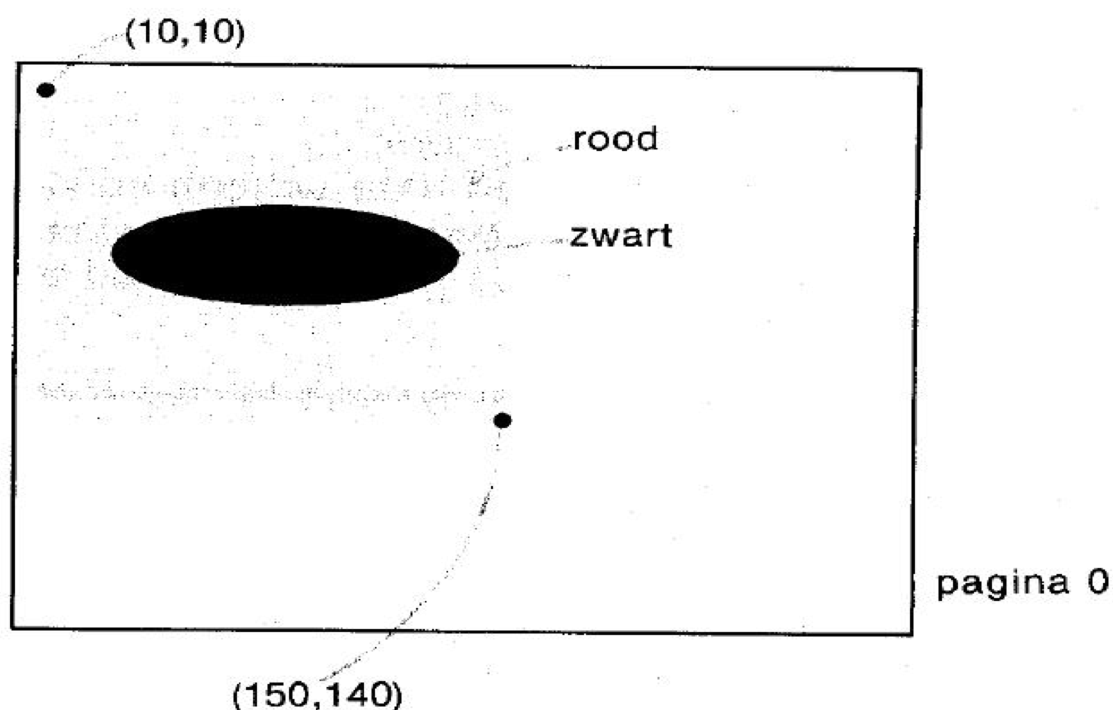
Tekeningen die gemaakt zijn met behulp van de SCREEN 5 en SCREEN 8 schermsoort kunnen gekopieerd worden. Bij kopiëren wordt er een gedeelte van het scherm bepaald en de kleurgegevens voor iedere stip binnen dat gebied worden op een andere plaats gekopieerd. Er zijn drie plaatsen waar dergelijke gegevens heen gekopieerd kunnen worden:

- het scherm (VRAM)
- het interne geheugen (een lijstvariabele)
- een diskette (bestand)

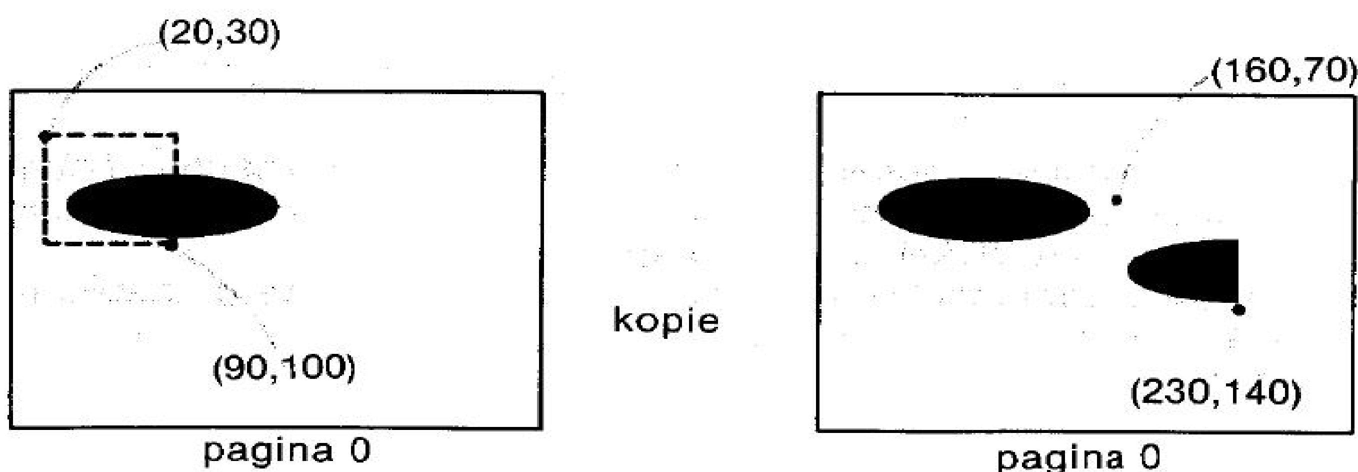
Het COPY bevel wordt gebruikt om grafische gegevens te kopiëren.

KOPIE VAN SCHERM NAAR SCHERM COPY (1)

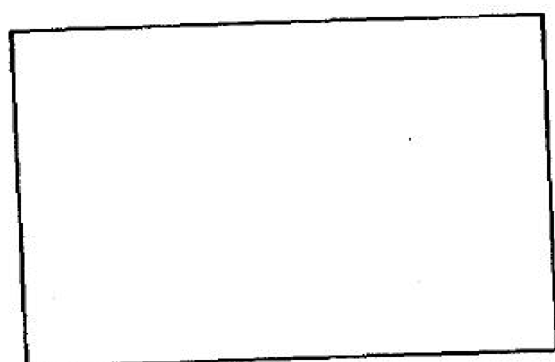
Bij het kopiëren van scherm naar scherm kun je op dezelfde of naar een andere pagina kopiëren. In beide gevallen worden de grafische gegevens in het VRAM van de computer gekopieerd.
Laten we bijvoorbeeld veronderstellen dat de volgende grafische figuren op pagina 0 van de SCREEN 5 beeldschermsoort getekend worden.



Wanneer het gedeelte van de grafische voorstelling van (20,30) t/m (90,100) (in de onderstaande afbeelding aangegeven door de stippellijn) op dezelfde pagina gekopieerd wordt in het gebied dat (160,70) als linker bovencoördinaat heeft, dan wordt het volgende effekt verkregen:

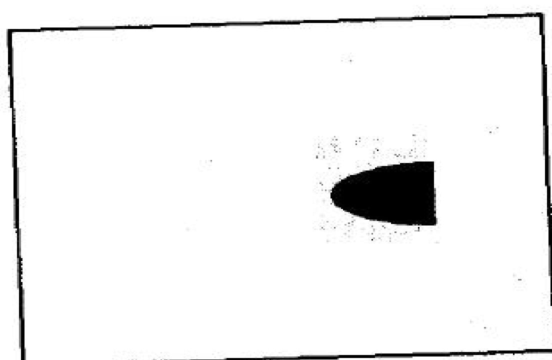


Je kunt ook naar een andere pagina kopiëren. Wanneer je hetzelfde gedeelte als dat in de bovenstaande afbeelding kopieert, dan krijg je het resultaat zoals dat in de onderstaande afbeelding wordt getoond.

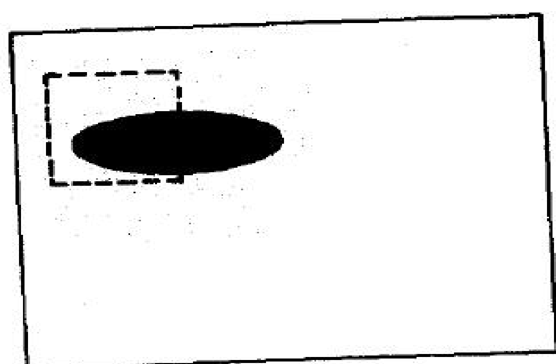


pagina 1

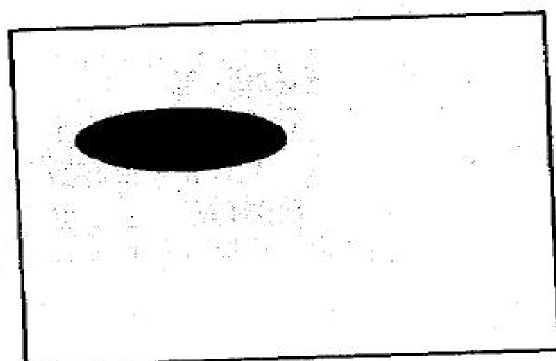
kopie



pagina 1



pagina 0



pagina 0

Bij het kopiëren worden de tekeningen en de kleuren op de voorgrond gekopieerd naar het gewenste gebied en de originele tekening en kleuren blijven op de originele pagina staan.
De schrijfwijze van het COPY bevel is als volgt:

COPY (X1,Y1)—(X2,Y2)[,te kopiëren pagina]TO(X3,Y3)[,bestemmingspagina]

(X1,Y1) is de linker boven coördinaat en (X2,Y2) is de rechter beneden coördinaat van het te kopiëren gebied. (X3,Y3) is de linker bovencoördinaat van het eindbestemmingsgebied. Als de te kopiëren pagina en/of de bestemmingspagina niet nader zijn aangegeven, wordt van de aktieve pagina uitgegaan.

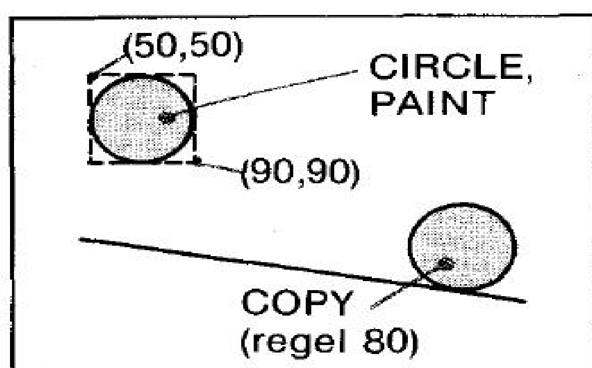
Je kunt dit aan de hand van het volgende programma kontroleren:

```
10 SCREEN 5
20 SET PAGE 0,0
30 LINE (10,10)-(150,140),8,BF
40 CIRCLE (90,60),40,1,,,.3
50 PAINT (90,60),1
60 COPY (20,30)-(90,100),0 TO (160,70),0
70 GOTO 70
```

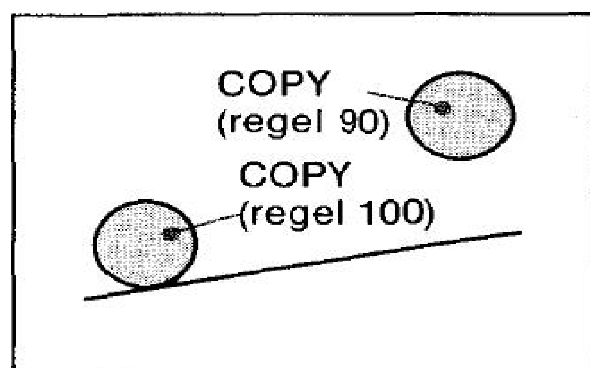
Regel 60 kopieert het gebied op pagina 0 van (20,30) t/m (90,100) naar het gebied, op dezelfde pagina, dat als linker bovencoördinaat (160,70) heeft.

```
10 SCREEN 5
20 SET PAGE 0,1:CLS
30 LINE (70,165)-(180,140),1
40 SET PAGE 1,0:CLS
50 LINE (70,140)-(180,165),1
60 CIRCLE (70,70),20,12
70 PAINT (70,70),12
80 COPY (50,50)-(90,90),0 TO (160,120),0
90 COPY (50,50)-(90,90),0 TO (160,50),1
100 COPY (50,50)-(90,90),0 TO (50,120),1
110 SET PAGE 0
120 FOR T=0 TO 300:NEXT T
130 SET PAGE 1
140 FOR T=0 TO 300:NEXT T
150 GOTO 110
```

Dit programma tekent de hieronder afgebeelde figuren op pagina 0 en pagina 1 bij de SCREEN 5 schermsoort. De eerste cirkel wordt met behulp van de CIRCLE en PAINT bevelen in regel 60 en 70 op pagina 0 getekend. De rest van de cirkels worden getekend door middel van het kopiëren met behulp van het COPY bevel (in de regels 80, 90 en 100).



pagina 0

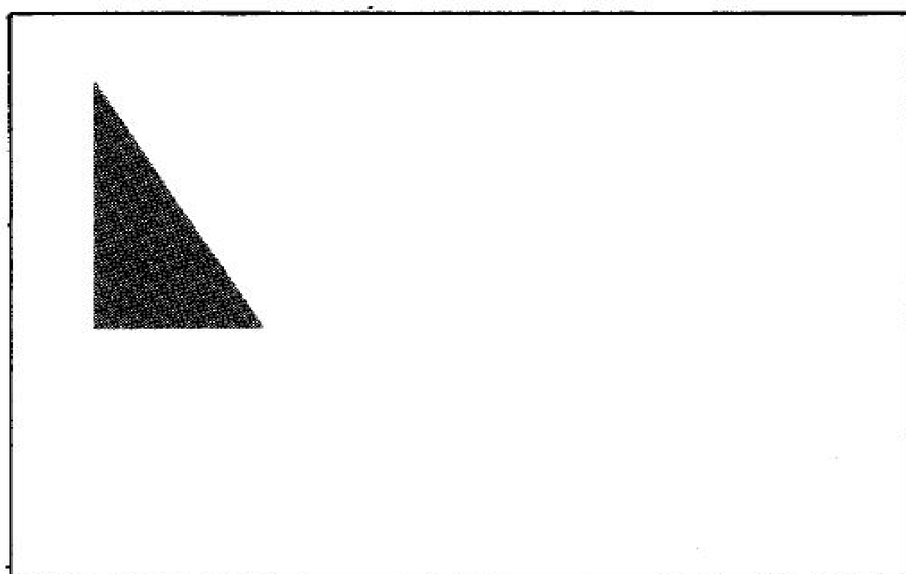


pagina 1

KOPIE VAN SCHERM NAAR INTERN GEHEUGEN EN VICE VERSA COPY (2)

Schermgegevens worden in het VRAM opgeslagen, maar kunnen ook naar het interne geheugen (RAM) gekopieerd worden. Je kunt eveneens gegevens die je in het interne geheugen hebt gekopieerd, weer van het interne geheugen naar het scherm (VRAM) kopiëren en de gegevens aldus opnieuw weergeven. Bij het terug kopiëren van het intern geheugen naar het VRAM (beeldscherm) kun je de richting van weergave (oriëntatie) op het beeldscherm desgewenst veranderen.

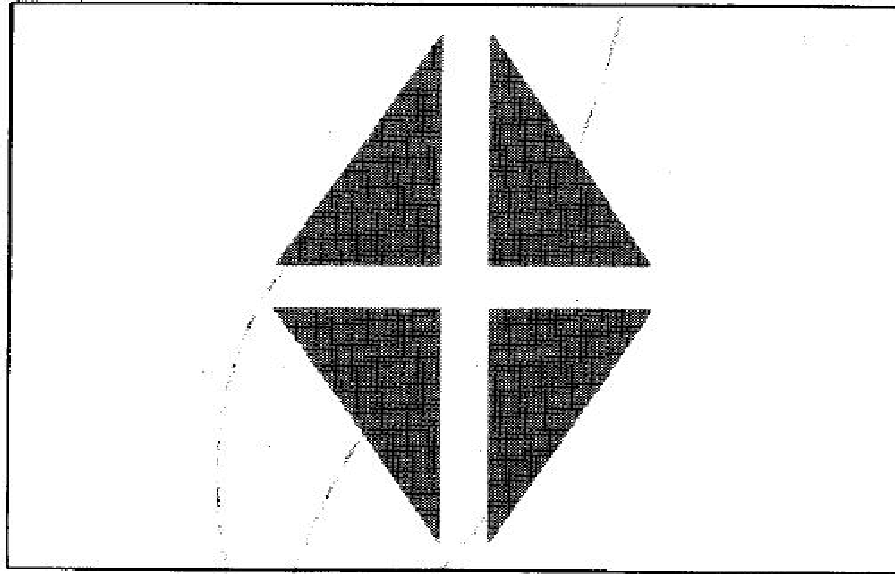
Laten we er bijvoorbeeld eens vanuit gaan dat je de volgende tekening op pagina 0 bij de SCREEN 5 schermsoort maakt.



pagina 0

Als je vervolgens het gebied op het beeldscherm van (20,20) t/m (60,105) naar het interne geheugen zou kopiëren, dan zou je de volgende weergave naar pagina 1 kunnen kopiëren.

Kopie waarbij de oriëntatie
onveranderd is



pagina 1

Kopie waarbij de oriëntatie
veranderd is

Alvorens grafische gegevens naar het interne geheugen te kopiëren, moet je eerst een numerieke lijstvariabele maken met behulp van het DIM bevel, om het geheugen in staat te stellen de gegevens te ontvangen.

De volgende schrijfwijze van het COPY bevel wordt gebruikt om grafische gegevens naar het geheugen te kopiëren.

COPY(X1,Y1)—(X2,Y2)[,te kopiëren pagina] TO naam lijstvariabele

(X1,Y1) is de linker bovencoördinaat en (X2,Y2) is de rechter benedencoördinaat van het te kopiëren gebied.
De grootte van de lijstvariabele wordt aan de hand van de volgende formule bepaald:

INT ((((ABS(X1 − X2) + 1)*(ABS(Y1 − Y2) + 1)*beeldpuntformaat
+ 7)/8 + 4)/8) + 1

INT en ABS zijn BASIC funkties die uitgelegd worden in hoofdstuk 7. Het beeldpuntformaat (het aantal beeldelementen dat gelijk staat aan een stip op het beeldscherm) varieert afhankelijk van de beeldschermsoort.

Beeldschermsoort	Beeldpuntformaat
SCREEN 5	4
SCREEN 6	2
SCREEN 7	4
SCREEN 8	8

Om bijvoorbeeld de gegevens van het gebied van (20,20) t/m (60,105) bij de SCREEN 5 schermsoort te kopiëren moet je voor X1 20, voor Y1 20, X2 60 en Y2 105 invullen. X1 – X2 = – 40 en Y1 – Y2 = – 85. Daarom zou de grootte van de lijstvariabele aan de hand van de eerder genoemde formule als volgt berekend worden:

```
INT((((ABS(-40)+1)*(ABS(-85)+1)*4+7)
/8+4)/8)+1
```

Dientengevolge zouden de volgende twee regels aan het begin van het programma geschreven moeten worden om de lijstvariabele P te bepalen.

```
S=INT((((ABS(-40)+1)*(ABS(-85)+1)
*4+7)/8+4)/8)+1
DIM P(S)
```

Wanneer eenmaal een dergelijke lijstvariabele bepaald is, kun je het COPY bevel gebruiken om de gegevens van het (20,20)—(60,105) gebied naar het geheugen—dat wil zeggen de lijstvariabele—te kopiëren.
In het COPY bevel hoef je de P variabele slechts op de volgende manier te schrijven:

```
COPY (20,20)-(60,105),0 TO P
```

Het volgende COPY bevel wordt gebruikt om gegevens van het geheugen (lijstvariabele) naar het scherm (VRAM) te kopiëren.

COPY lijstvariabele [,oriëntatie] TO(X3,Y3) [,bestemmingspagina]]

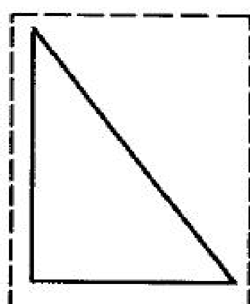
(X3,Y3) vormt het beginpunt op de bestemmingspagina voor het weergeven van de te kopiëren gegevens:

Er is sprake van vier "oriëntaties", die je bepaalt door op één van de cijfers 0 t/m 3. "Oriëntatie" geeft de richting aan waarin de gegevens moeten worden weergegeven vanaf het beginpunt.

"Orientatie"-nummer	Tekenrichting
0	van linksboven naar rechtsonder ↘
1	van rechtsboven naar linksonder ↙
2	van linksonder naar rechtsboven ↗
3	van rechtsonder naar linksboven ↖

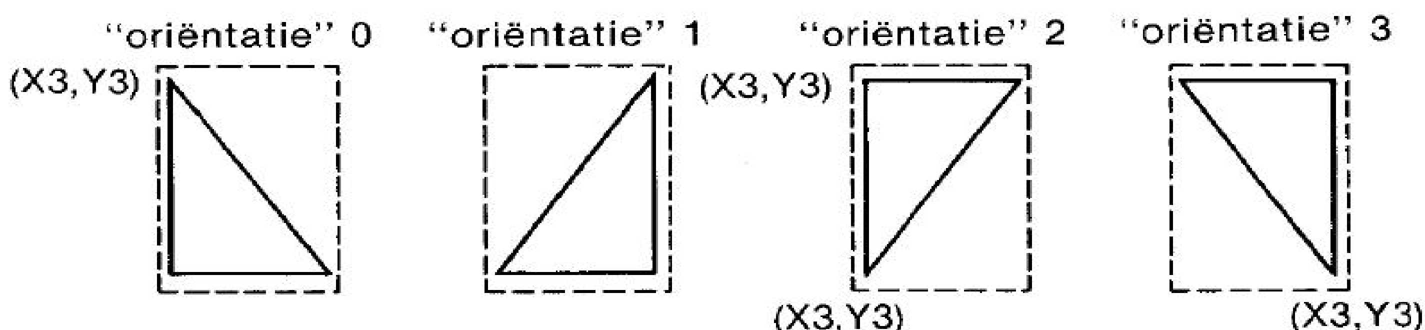
Bij weglaten van het oriëntatienummer wordt ingesteld op 0.

Wanneer bijvoorbeeld de gegevens voor de volgende tekening naar het geheugen zijn gekopieerd:



Het gebied binnen de stippellijnen wordt naar het geheugen gekopieerd

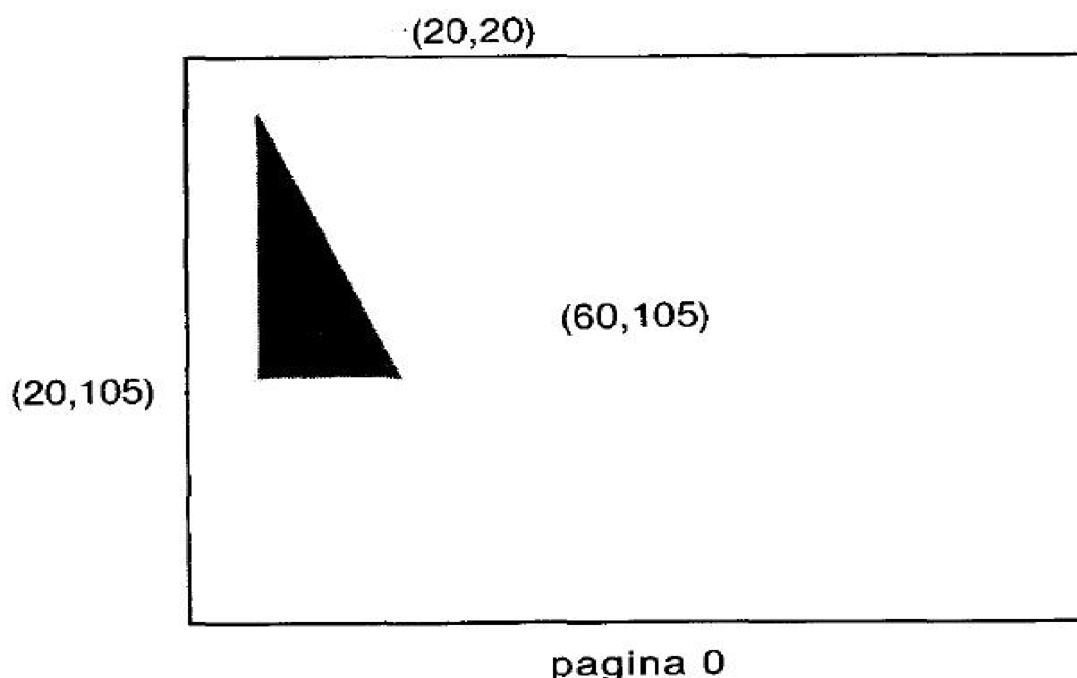
dan kunnen deze gegevens naar het scherm terug worden gekopieerd. Gebruik hiervoor de vier oriëntatienummers en de (X3,Y3) beginpunten voor het tekenen als volgt:



Bij weglaten van het oriëntatienummer wordt ingesteld op 0, van links boven naar rechts beneden.
Laten we een programma schrijven dat het voorbeeld op blz. 130 weergeeft.

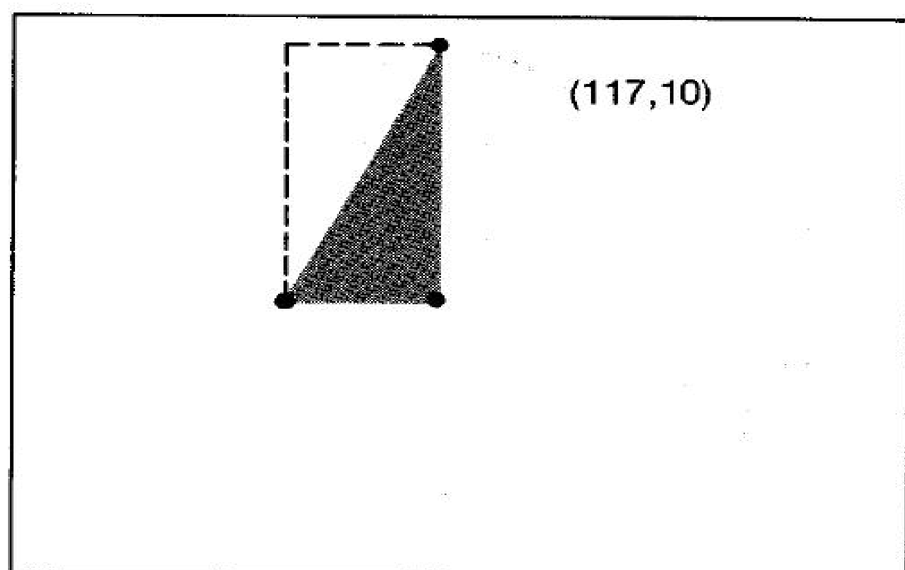
```
10 S=INT((((ABS(-40)+1)*(ABS(-85)+1)*4+7
)/8+4)/8)+1
20 DIM P(S)
30 SCREEN 5
40 SET PAGE 0,1:CLS
50 SET PAGE 0,0:CLS
60 LINE (20,20)-(20,105)
70 LINE (20,20)-(60,105)
80 LINE (20,105)-(60,105)
90 PAINT (25,50)
100 COPY (20,20)-(60,105),0 TO P
110 COPY P,0 TO (138,10),1
120 COPY P,1 TO (117,10),1
130 COPY P,2 TO (138,200),1
140 COPY P,3 TO (117,200),1
150 SET PAGE 1
160 GOTO 160
```

De lijstvariabele P wordt in de regels 10 en 20 bepaald. Vervolgens wordt de volgende tekening op pagina 0 gemaakt op het SCREEN 5 scherm.



pagina 0

De COPY bevel in regel 100 kopieert het gebied van (20,20) t/m (60,105)—waarbinnen de tekening zich bevindt—naar de lijstvariabele in het geheugen. Vervolgens veranderen de regels 110 t/m 140 de weergaverichting van de gegevens die naar pagina 1 in het geheugen zijn gekopieerd.
Regel 120 bijvoorbeeld kopieert de gegevens met gebruikmaking van "oriëntatie" 1 en begint op het punt (117,10) met tekenen.



pagina 1

De andere COPY bevelen vervullen soortgelijke funkties en maken een tekening, zoals die op blz. 130, op pagina 1.

KOPIE VAN SCHERM NAAR DISKETTE EN VICE VERSA COPY (3)

Het COPY bevel kan ook gebruikt worden om gegevens die op het scherm (VRAM) getekend zijn naar een diskettebestand te kopiëren. De computer moet dan echter wel voorzien zijn van een interne diskette-eenheid of er moet een externe diskette-eenheid aangesloten zijn.
De schrijfwijze van het COPY bevel voor kopiëren (opslaan) van grafische gegevens op diskette is:

COPY (X1,Y1)—(X2,Y2) [, te kopiëren pagina] TO "[naam diskette-eenheid] bestandsnaam [.soortnaam]"

De regels voor het toewijzen van een bestandsnaam aan gegevens zijn precies hetzelfde als die voor bestandsnamen bij het opslaan van programma's. De soortnaam kun je desgewenst weglaten, maar het verdient aanbeveling deze altijd in te vullen, zodat je weet om wat voor een soort bestand het gaat. In de volgende programmavoorbeelden wordt als soortnaam .TEK gebruikt.
Laten we nu een programma schrijven om de tekening die we eerder naar het geheugen hebben gekopieerd, nu naar de diskette te kopiëren.

```
10 SCREEN 5
20 SET PAGE 0,0:CLS
30 LINE (20,20)-(20,105)
40 LINE (20,20)-(60,105)
50 LINE (20,105)-(60,105)
60 PAINT (25,50)
70 COPY (20,20)-(60,105),0 TO "DRIEHOEK.
TEK"
```

Door de regels 30 t/m 60 wordt er een driehoek getrokken. Vervolgens kopieert regel 70 het gebied van (20,20)—(60,105), waarbinnen de driehoek zich bevindt, naar de diskette. De bestandsnaam is DRIEHOEK en de soortnaam is .TEK.
Bij uitvoering van dit programma wordt er op pagina 0 van het SCREEN 5 scherm een driehoek getrokken. Direkt nadat de driehoek getrokken is, begint de diskette-eenheid te werken en de gegevens van de tekening worden naar de diskette gekopieerd.
Wanneer de gegevens gekopieerd zijn, eindigt het programma en op het scherm verschijnt het "Ok" bericht.
Bij uitvoering van het FILES bevel verschijnen de bestands- en soortnaam

```
DRIEHOEK.TEK
```

op het scherm en kun je kontroleren of de gegevens van de tekening inderdaad naar de diskette zijn gekopieerd.

De schrijfwijze van het COPY bevel om grafische gegevens van de diskette weer naar het scherm (VRAM) te kopiëren is:

COPY "[naam diskette-eenheid] bestandsnaam [.soortnaam]" [, oriëntatie] TO (X3,Y3) [, bestemmingspagina]

Hierbij worden dezelfde bestands- en soortnaam ingevuld als dat het geval was bij het kopiëren van gegevens van het scherm naar de diskette. De "oriëntatie" en de coördinaten voor het beginnen met tekenen zijn precies dezelfde als die gebruikt werden voor het kopiëren vanuit het geheugen.
Het volgende programma verandert de oriëntatie van de op de diskette gekopieerde gegevens in vier verschillende oriëntaties. En kopieert vervolgens de gegevens weer naar het scherm.

```
10 SCREEN 5
20 SET PAGE 1,1:CLS
30 COPY "DRIEHOEK.TEK",0 TO (138,10),1
40 COPY "DRIEHOEK.TEK",1 TO (117,10),1
50 COPY "DRIEHOEK.TEK",2 TO (138,200),1
60 COPY "DRIEHOEK.TEK",3 TO (117,200),1
70 GOTO 70
```

De COPY bevelen in de regels 30 t/m 60 kopiëren de tekening naar pagina 1 bij de SCREEN 5 schermsoort, waarbij er sprake is van verschillende oriëntaties en verschillende beginpunten voor het maken van de tekening. Als je dit programma uitvoert dan zul je zien hoe de gegevens gekopieerd worden.

In het bovenstaande programma waren de aktieve en de weergegeven pagina ingesteld op pagina 1. Je kunt echter ook de aktieve en de weergegeven pagina op verschillende pagina's instellen. Wanneer je in dit geval de gegevens eerst van de diskette naar de aktieve pagina kopieert en, na het voltooien van de tekening, de aktieve pagina in de weergegeven pagina verandert, dan zal de voltooide tekening in één keer op het scherm verschijnen.
Herzie het voorgaande programma als volgt en voer het uit.

```
10 SCREEN 5
20 SET PAGE 0,1:CLS←—— veranderd
30 COPY "DRIEHOEK.TEK",0 TO (138,10),1
40 COPY "DRIEHOEK.TEK",1 TO (117,10),1
50 COPY "DRIEHOEK.TEK",2 TO (138,200),1
60 COPY "DRIEHOEK.TEK",3 TO (117,200),1
65 SET PAGE 1←——toegevoegd
70 GOTO 70
```

KOPIE VAN GEHEUGEN NAAR DISKETTE EN VICE VERSA COPY (4)

De schrijfwijze voor het COPY bevel om grafische gegevens die op een diskette zijn opgeslagen, naar een lijstvariabele in het geheugen te kopiëren, is als volgt:

COPY "[naam diskette-eenheid] bestandsnaam [. soortnaam]" TO naam lijstvariabele

Maak gebruik van de volgende schrijwijze om grafische gegevens van een lijstvariabele in het geheugen naar een diskette te kopiëren:

COPY naam lijstvariabele TO "[naam diskette-eenheid] bestandsnaam [. soortnaam]"

LOGISCHE BEWERKINGEN

Bij het kopiëren van grafische gegevens met behulp van het COPY bevel, kunnen er logische bewerkingen worden uitgevoerd tussen de kleurcode van de tekening en de kleurcode van het bestemmingsscherm.
De volgende tien logische bewerkingen kunnen bij het COPY bevel gebruikt worden.

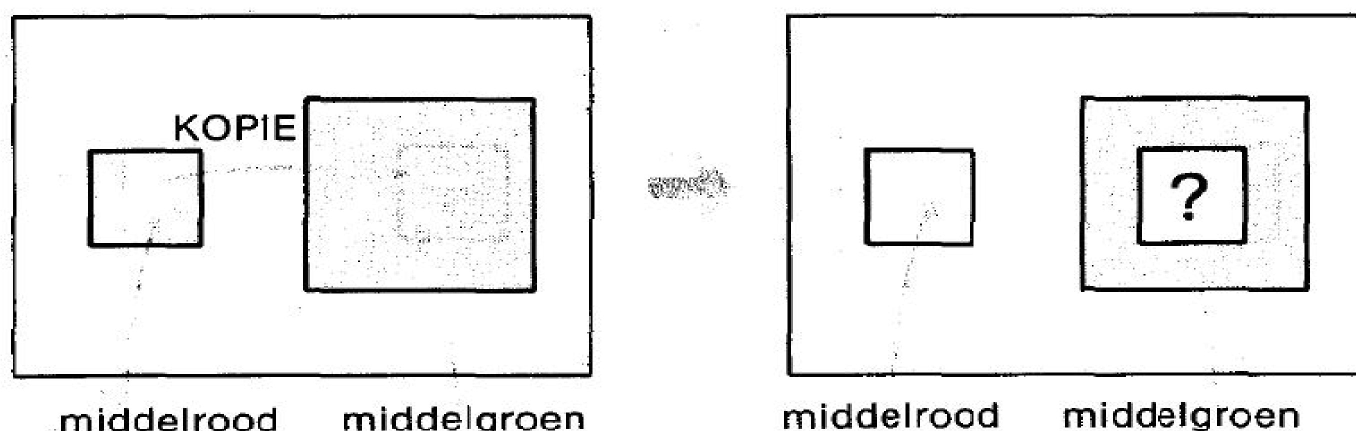
PSET, PRESET, XOR, OR, AND

TPSET, TPRESET, TXOR, TOR, TAND

Het resultaat van de logische bewerking kun je zien wanneer de kleurcodes in binaire of tweetallige codes veranderd worden. De volgende tabel toont de binaire kleurcodes van de 16 kleuren van het SCREEN 5 scherm bij starten van BASIC.

Kleur	Kleurcode (decimaal)	Kleurcode (binair)	Kleur	Kleurcode (decimaal)	Kleurcode (binair)
transparant	0	0000	middelrood	8	1000
zwart	1	0001	lichtrood	9	1001
middelgroen	2	0010	donkergeel	10	1010
lichtgroen	3	0011	lichtgeel	11	1011
donkerblauw	4	0100	donkergroen	12	1100
lichtblauw	5	0101	roodpaars	13	1101
donkerrood	6	0110	grijs	14	1110
hemelsblauw	7	0111	wit	15	1111

Laten we eens nagaan wat er gebeurt wanneer we een middelrode figuur (kleurcode 1000) op een middelgroene figuur (kleurcode 0010) kopiëren.



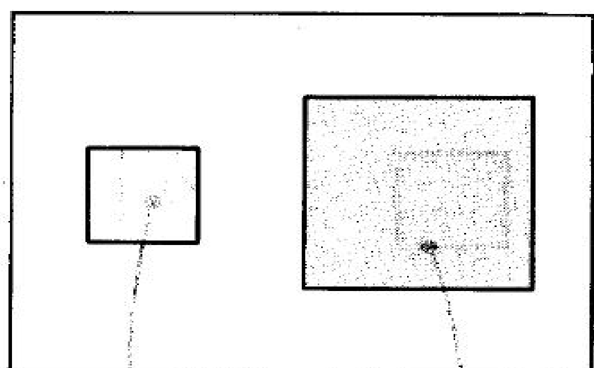
Wanneer er geen logische bewerking wordt uitgevoerd, wordt het middelrode vierkant over het middelgroene vierkant heen weergegeven. Als er echter een logische bewerking wordt uitgevoerd, wordt het gedeelte van het middelgroene vierkant dat in de afbeelding met [?] staat aangegeven een andere kleur dan middelrood. De kleur van dit gedeelte is afhankelijk van welke logische bewerking er uitgevoerd wordt. Bij wijze van voorbeeld nemen we de logische bewerking OR. OR voert de volgende bewerkingen met ieder cijfer (0 of 1) van de twee binaire kleurcodes uit.

X	Y	resultaat van X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

In het bovenstaande voorbeeld wordt de OR bewerking met ieder paar binaire cijfers van de middelrode kleurcode 1000 en de binaire kleurcodecijfers van middelgroen (0010) uitgevoerd, zoals hieronder getoond wordt.

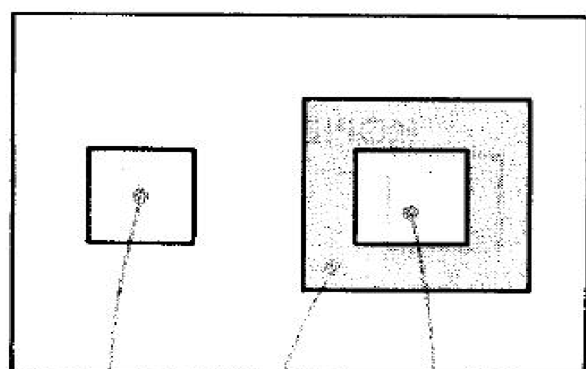
middelrood........1000
middelgroen.....0010
OR.................................1010

Het resultaat van de OR bewerking is 1010, hetgeen—zoals in de bovenstaande kleurentabel aangegeven wordt—donkergeel is. Wanneer er dus een middelrode figuur op een middelgroene figuur met behulp van de OR logische bewerking gekopieerd wordt, wordt deze figuur donkergeel.



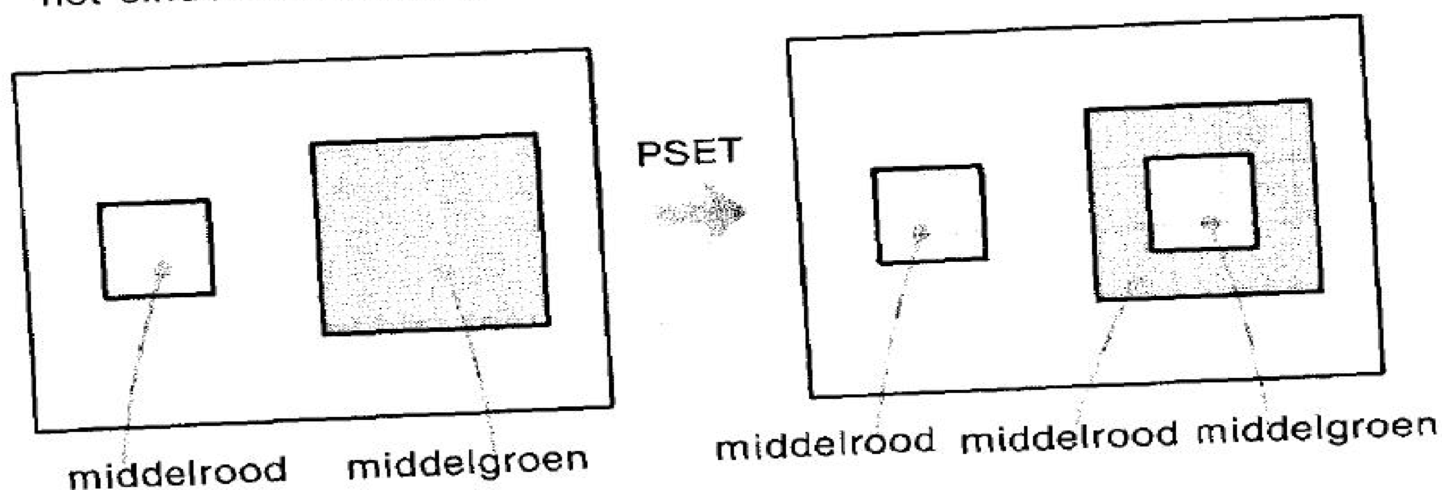
geko-
pieerd
m.b.v.
OR



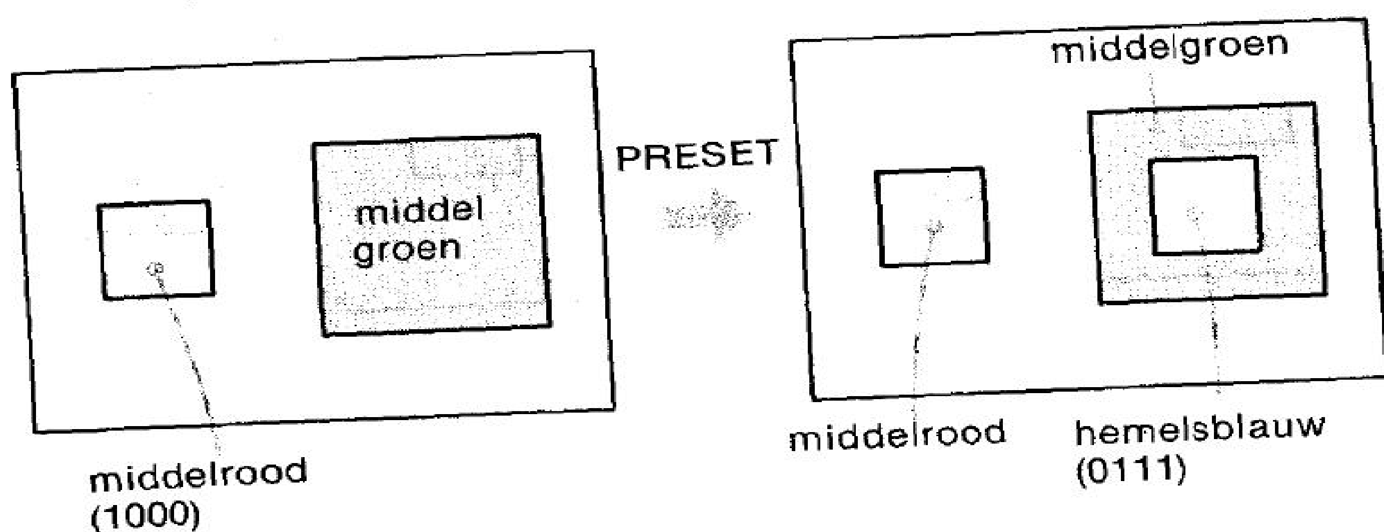


De resultaten van andere logische bewerkingen dan de OR bewerking staan hieronder.

- PSET—de gekopieerde kleur blijft hetzelfde, ongeacht de kleur van het eindbestemmingsgebied.



- PRESET—verandert ieder kleurcodecijfer van de te kopiëren kleur in het tegenovergestelde cijfer—0 in 1 en 1 in 0—, waarbij er geen rekening wordt gehouden met de kleur van het eindbestemmingsgebied. Als je bijvoorbeeld middelrood (1000) met behulp van PRESET kopieert, dan verandert de kleur in hemelsblauw (0111).

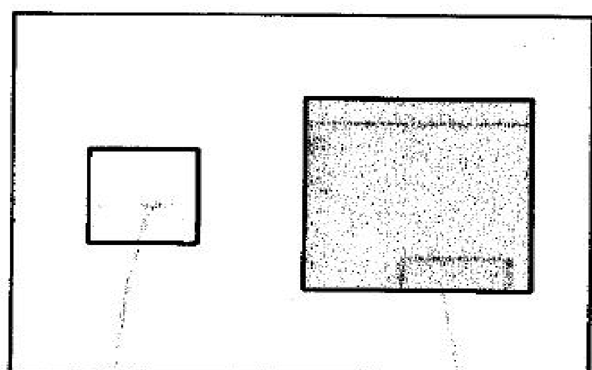


- XOR—hierbij worden de volgende bewerkingen met de kleurcode van de te kopiëren kleur en de kleurcode van het eindbestemmingsgebied uitgevoerd.

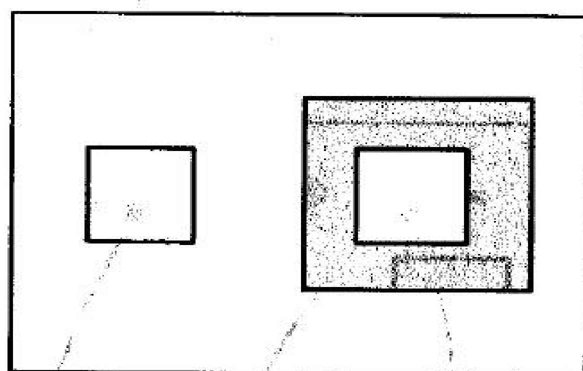
X	Y	resultaat van X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Wanneer de XOR bewerking bijvoorbeeld met middelrood (1000) en middelgroen (0010) wordt uitgevoerd, dan is het resultaat donkergeel (1010).

1000
<u>0010</u>
1010



XOR



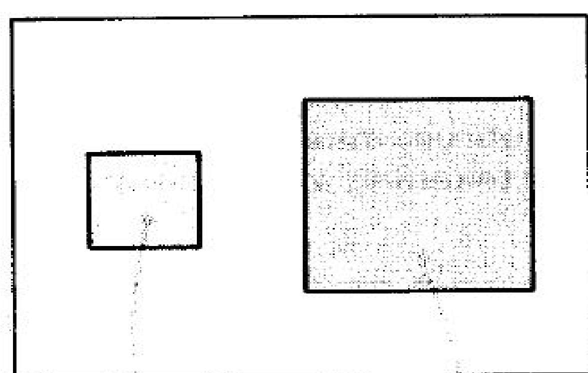
middelrood (1000) middelgroen (0010) middelrood middel groen donkergeel (1010)

● AND—voert de volgende bewerking met de kleurcode van de te kopiëren kleur en de kleurcode van het eindbestemmingsgebied uit.

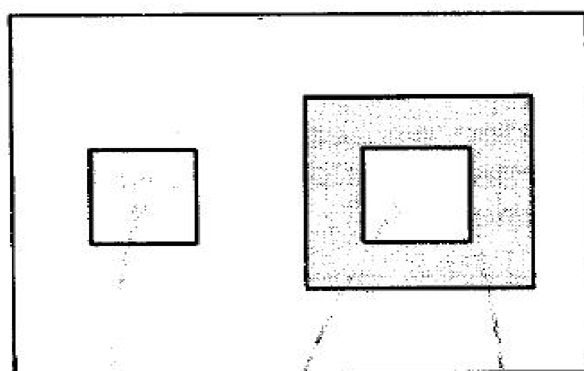
X	Y	resultaat van X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Bij het uitvoeren van de AND bewerking met middelrood (1000) en middelgroen (0010), dan is het resultaat de transparante kleur (0000).

1000
0010
0000



AND



● TPSET, TPRESET, TOR, TXOR, TAND—deze bewerkingen zijn vrijwel hetzelfde als PSET, PRESET, OR XOR en AND. Het verschil is dat wanneer er sprake is van de transparante kleur en er T voor deze bewerkingen wordt gezet, de kleur na het kopiëren transparant blijft, ongeacht de kleur van het eindbestemmingsgebied.

Gebruik van logische bewerkingen bij het COPY bevel
Bij gebruik van logische bewerkingen is de schrijfwijze van het COPY bevel als volgt:

● **Van scherm naar scherm**

COPY (X1,Y1)—(X2,Y2) [, te kopiëren pagina] TO (X3,Y3) [, bestemmingspagina], [, **logische bewerking**]

Voorbeeld: COPY (10,10)—(100,100), 0 TO (30,30), 1, XOR

● **Van geheugen (lijstvariabele) naar scherm**

COPY naam lijstvariabele name [, orientatie] TO (X3,Y3) [, bestemmingspagina], [, **logische bewerking**]

Voorbeeld: COPY P, 1 TO (30,30), 0, TAND

● **Van diskette (bestand) naar scherm**

COPY "[naam diskette-eenheid] bestandsnaam [. soortnaam]" [, oriëntatie] TO (X3,Y3) [, bestemmingspagina] [, **logische bewerking]**

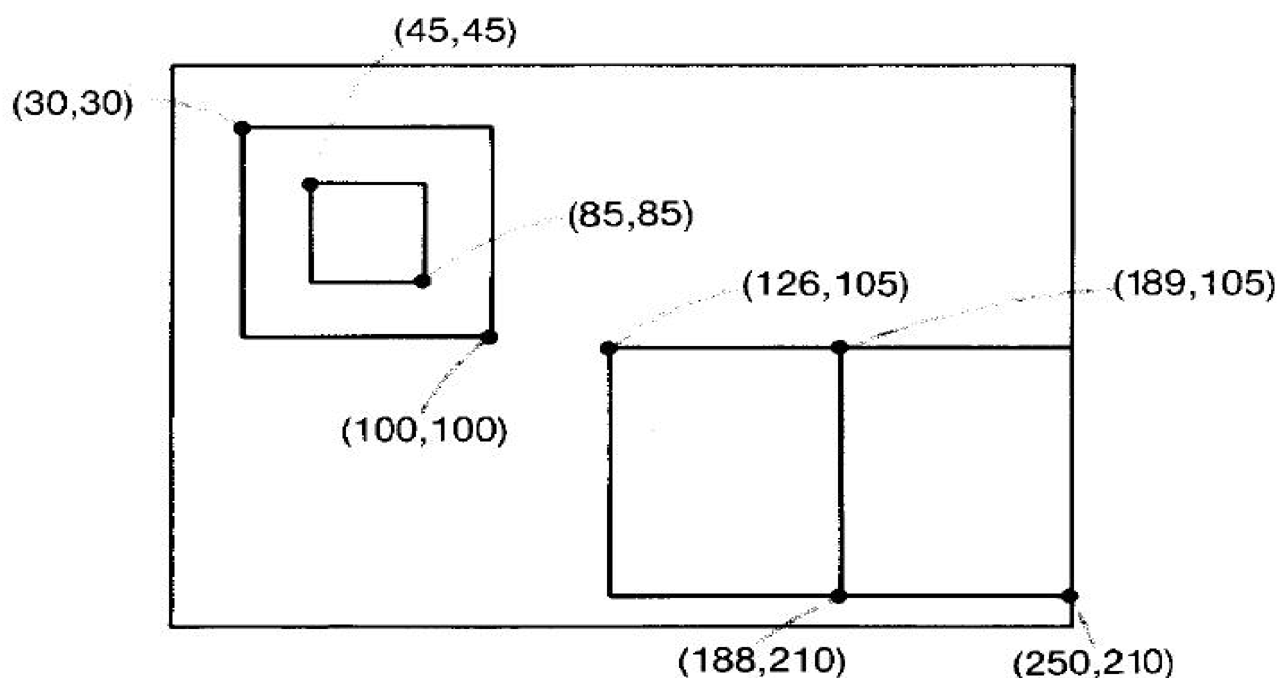
Voorbeeld: COPY "DRIEHOEK .TEK", 2 TO (30,30), 1, PRESET

Wanneer je de logische bewerking weglaat, dan is het effekt hetzelfde als wanneer je PSET invult.

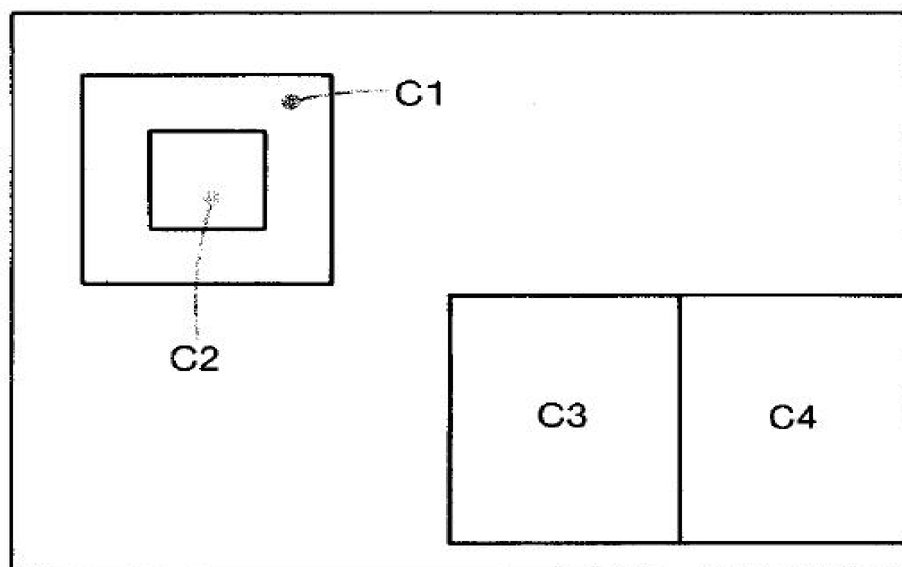
Laten we een programma schrijven dat gebruik maakt van een logische bewerking bij het kopiëren van een tekening van scherm naar scherm.

```
10 SCREEN 5
20 SET PAGE 0,0:CLS
30 READ C1,C2,C3,C4
40 LINE (30,30)-(100,100),C1,BF
50 LINE (45,45)-(85,85),C2,BF
60 LINE (126,105)-(188,210),C3,BF
70 LINE (189,105)-(250,210),C4,BF
80 COPY (30,30)-(100,100),0 TO (153,124)
,0,OR
90 GOTO 90
100 DATA 8,3,10,2
```

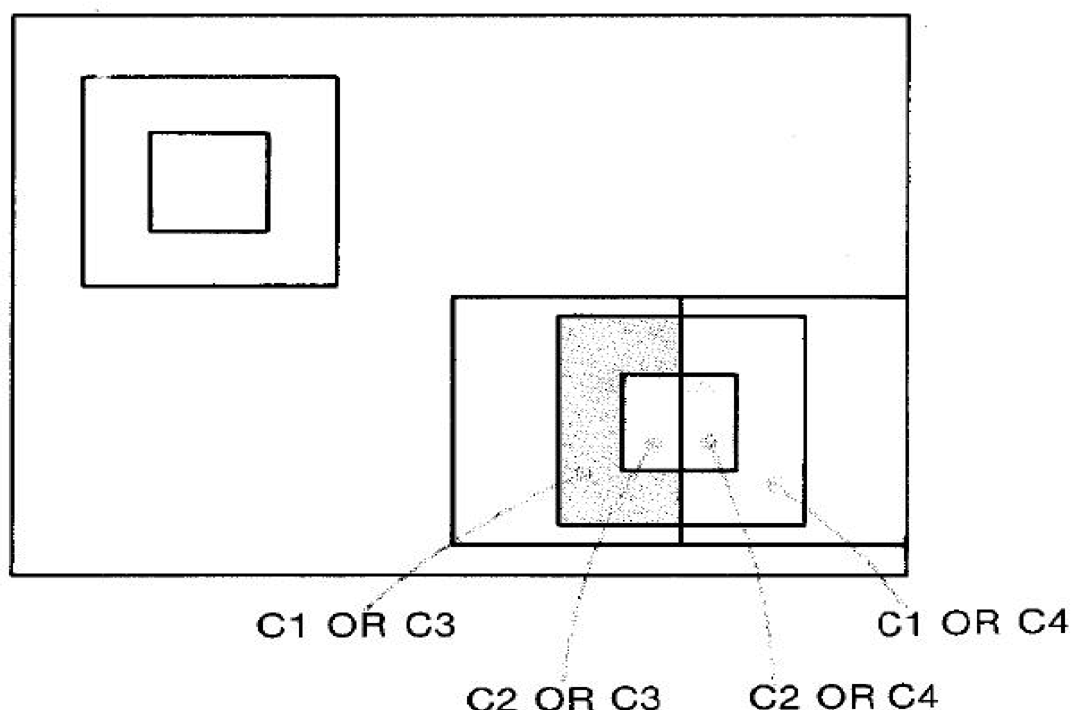
Dit programma maakt de volgende tekening.



De kleuren worden bepaald door de waarden die toegekend worden aan de C1, C2, C3 en C4 variabelen.

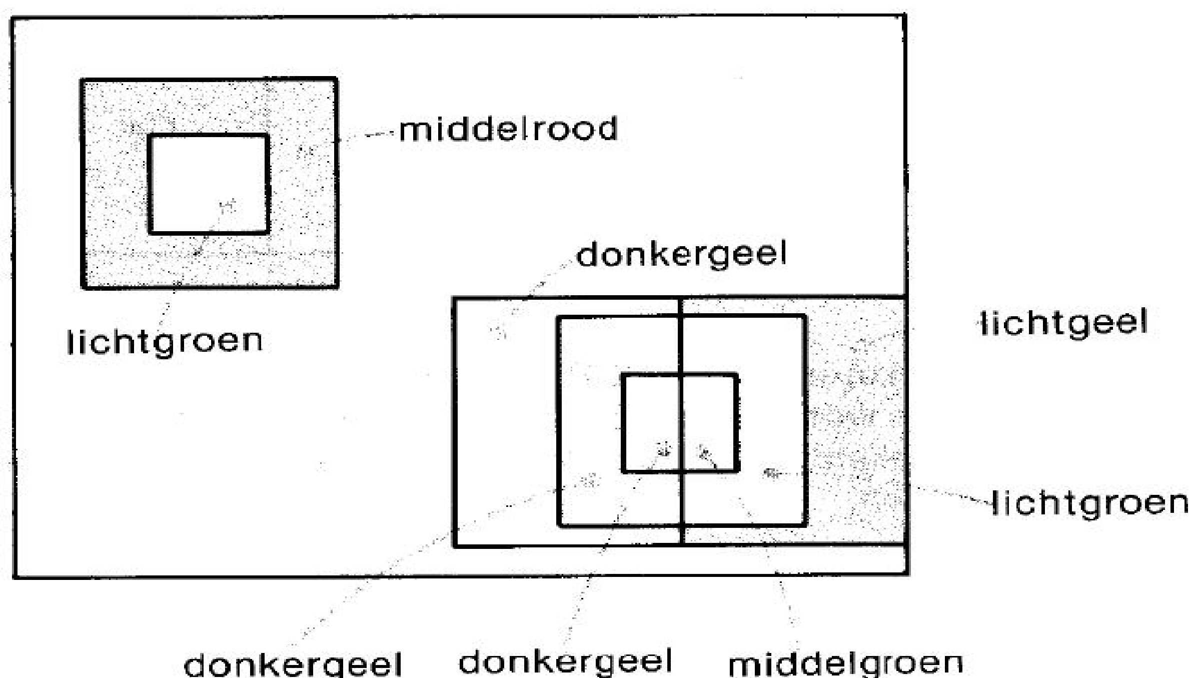


Vervolgens kopieert het COPY bevel in regel 80 de tekening. Tijdens het kopiëren worden er logische bewerkingen uitgevoerd met die delen van de tekening waar er twee tekeningen over elkaar heen weergegeven worden.



Dit programma toont de resultaten van logische bewerkingen die vier verschillende kleuren kombineren. De kleur die aan de verschillende variabelen in het programma toegewezen zijn, bestaan uit: C1, middelrood; C2 lichtgroen; C3 donkergeel en C4 middelgroen (het DATA bevel in regel 100). De logische bewerking die uitgevoerd wordt is OR (regel 80).

Uitvoering van het programma resulteert in:



HET SCREEN BEVEL

- Instellingen met het SCREEN bevel
- Intoetssignaal aan/uit
- Overdrachtssnelheid van/naar cassetteband
- Soort afdrukker
- Vervlechting

DE SCREEN INSTELLINGEN

Het SCREEN bevel wordt gebruikt om een aantal instellingen te maken, in aanvulling op de instelling op een schermsoort.

SCREEN [schermsoort], [formaat beeldpatroon], [intoetssignaal aan/uit], [overdrachtssnelheid], [soort afdrukker], [vervlechting]

INTOETSSIGNAAL, OVERDRACHTSSNELHEID, SOORT AFDRUKKER

Intoetssignaal aan/uit
De derde parameter van het SCREEN bevel (intoetssignaal aan/uit) bepaalt of het indrukken van een toets op het toetsenbord al dan niet een geluid veroorzaakt. Wanneer intoetssignaal aan/uit op 0 wordt ingesteld, dan veroorzaakt het indrukken van een toets geen geluid. Bij het instellen op een van de andere cijfers (1—255) is er sprake van een klikgeluid bij indrukken van een toets.

```
SCREEN ,,0 —— intoetssignaal uit
SCREEN ,,1 —— intoetssignaal aan
```

Instelling overdrachtssnelheid van/naar cassetteband
De overdrachtssnelheid is het aantal bits per sekonde waarmee gegevens worden overgezonden. Een overdrachtssnelheid van 1200 betekent dat er per sekonde 1200 bits aan gegevens worden overgezonden. Bij een overdrachtssnelheid van 2400 worden er 2400 bits aan gegevens per sekonde overgezonden.
De vierde parameter van het SCREEN bevel bepaalt de overdrachtssnelheid voor het overzenden van gegevens van en naar cassetteband. Overdrachtssnelheid 1 is 1200 bits en 2 is 2400 bits. De oorspronkelijke instelling is 1 (1200 bits).

```
SCREEN ,,,1 —— 1200 baud (bits/sek.)
SCREEN ,,,2 —— 2400 baud (bits/sek.)
```

Als je met behulp van het SCREEN ,,,2 bevel de overdrachtssnelheid ingesteld hebt op 2400 baud alvorens een programma op cassetteband op te slaan, dan zul je weer met behulp van het SCREEN ,,,2 bevel op 2400 baud moeten instellen om het programma van de cassetteband te kunnen laden.

Soort afdrukker
De vijfde parameter van het SCREEN bevel dient voor de aanpassing aan het gebruikte type printer of afdrukeenheid. 0 is de instelling voor MSX-type afdrukkers. Voor andere soorten afdrukkers zijn de instellingen 1 t/m 255 beschikbaar. De oorspronkelijke instelling, die geldt bij het inschakelen, is 0 (voor MSX-type afdrukkers).
De MSX-type afdrukkers zijn speciaal ontworpen voor gebruik met MSX-computers en beschikken over een repertoire aan lettertekens dat de specifieke MSX grafische tekens omvat.

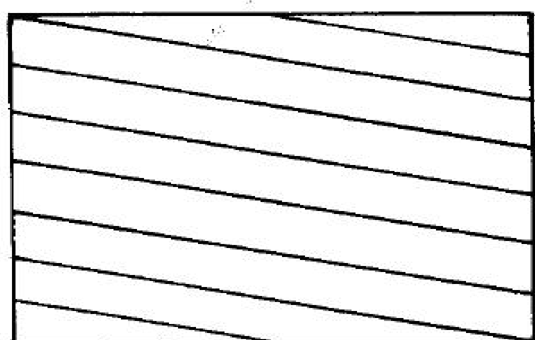
Als een ander soort afdrukker dan het MSX-type wordt gebruikt, kunnen na het SCREEN ,,,,1 bevel de speciale MSX grafische tekens niet weergegeven worden, zodat er in plaats daarvan spaties open blijven.

DE VERVLECHTINGSFUNKTIE

Beeldscherm aftasten met vervlechting
Gewoonlijk vindt de beeldscherm-aftasting bij MSX-computers plaats zonder vervlechting van beeldlijnen. De vervlechtingsfunktie kan echter ingeschakeld worden met de zesde parameter (vervlechtingsfunktie) van het SCREEN bevel.

Bij aftasten met vervlechting zijn de plaatsen van het eerst afgetaste veld en het als tweede afgetaste veld verschillend. Dit maakt een fijnere detaillering van de weergave mogelijk. Bij aftasten met de vervlechtingsfunktie is het wel nodig een TV of videomonitor met nagloeiend scherm te gebruiken, aangezien het beeld bij een TV of monitor met normaal scherm kan gaan flikkeren.

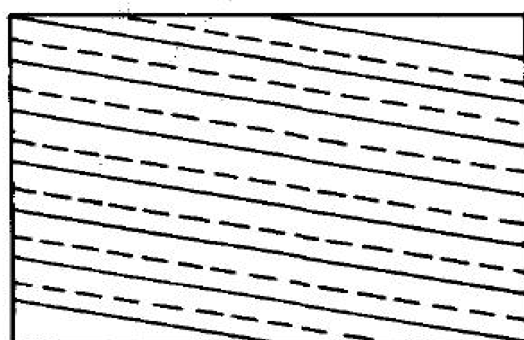
Het eerste veld en het tweede veld nemen dezelfde plaats in.



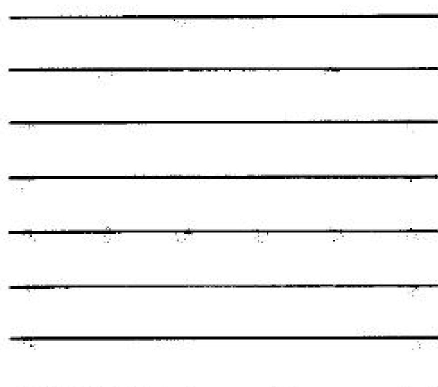
Aftasten zonder vervlechting

eerste veld

tweede veld



Aftasten met vervlechting



Aftasten zonder vervlechting



Aftasten met vervlechting

Weergave met afwisselend even/oneven pagina's

De parameter voor het inschakelen van de vervlechtingsfunktie kan ook gebruikt worden voor het instellen van weergave met afwisselend even en oneven pagina's. Hiervoor moet eerst de weergegeven pagina oneven (1 of 3) gemaakt worden met de PAGE instruktie. Dan worden na het kiezen van de weergave met afwisselende pagina's, met behulp van het SCREEN bevel, de weergegeven pagina en voorgaande pagina om-en-om, in hoog tempo afwisselend, op het scherm afgebeeld. (Met de voorgaande pagina bedoelen we de pagina waarvan het nummer één kleiner is dan dat van de weergegeven pagina.)
De volgende tabel toont de instellingen van de vervlechtingsfunktie.

Vervlechting	Ingestelde funktie
0	normaal (geen vervlechting, geen afwisselen van pagina's)
1	vervlechtingsfunktie
2	weergave met afwisselend even en oneven pagina's
3	weergave met afwisselend even en oneven pagina's en vervlechting

Het volgende programma tekent een gele cirkel op pagina 0 en een witte ovaal op pagina 1 bij de SCREEN 5 beeldschermsoort. Eerst worden pagina 0 en pagina 1 afwisselend afgebeeld, doordat de weergegeven pagina gewijzigd is met het SET PAGE bevel, waarbij het tempo van de wisseling steeds sneller wordt. Vervolgens wordt een interessant effekt bereikt doordat aan het einde van het programma wordt overgegaan op weergave met afwisselend even en oneven pagina's.

```
10 COLOR 15,4,4:SCREEN 5,,,,,0
20 SET PAGE 0,1:CLS
30 XC=128 : YC=100
40 '*** gele cirkel ***
50 SET PAGE 0,0
60 CIRCLE (XC,YC),45,10
70 PAINT (XC,YC),10
80 FOR T=0 TO 2000:NEXT T
90 '*** witte ovaal
100 SET PAGE 1,1
110 CIRCLE (XC,YC),90,15,,,.7
120 PAINT (XC,YC),15
130 FOR T=0 TO 2000:NEXT T
140 '*** pagina's wisselen ***
150 J=1200:DP=0:AP=1
160 SET PAGE DP,DP:SWAP DP,AP
170 FOR T=0 TO J:NEXT T
180 J=J*.8:IF J>1 THEN 160
190 '*** even/oneven weergave
200 SET PAGE 1,1
210 SCREEN ,,,,,2
220 GOTO 220
```

Hoofdstuk 6
Beeldpatronen (Sprites)

SAMENSTELLING EN GEBRUIK VAN BEELDPATRONEN

- Beeldpatronen (sprites)
- Samenstellen van een beeldpatroon—SPRITE$
- Weergeven van een beeldpatroon—PUT SPRITE
- Bewegen van beeldpatronen

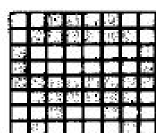
BEELDPATRONEN (SPRITES)

Een sprite is een vrij naar wens samen te stellen beeldpatroon van 8×8 of 16×16 beeldpunten of stippen, dat over het scherm bewegen kan. MSX2-BASIC biedt de mogelijkheid beeldpatronen weer te geven en te laten bewegen op 32 verschillende beeldvlakken.

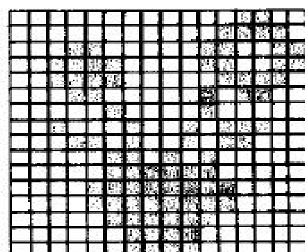
Beeldschermsoort en beeldpatronen
Beeldpatronen kunnen gebruikt worden met alle beeldschermsoorten van SCREEN 1 t/m SCREEN 8, oftewel met alle schermen behalve SCREEN 0. Beeldpatronen met de schermen SCREEN 4 t/m SCREEN 8 hebben diverse toepassingen die met beeldschermsoorten SCREEN 1 t/m 3 niet beschikbaar zijn.

Soorten beeldpatronen
Elk afzonderlijk beeldpatroon bestaat uit 8×8 of 16×16 beeldpunten, ook stippen genoemd. Een beeldpatroon kan zowel in standaardformaat of in vergroot formaat op het scherm afgebeeld worden. De afmetingen van een beeldpatroon in vergroot formaat bedragen zowel horizontaal als vertikaal tweemaal die van het standaard formaat.

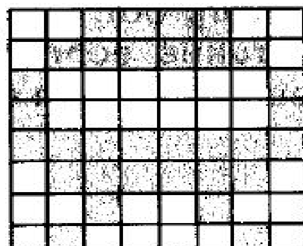
×8 stippen,
tandaard



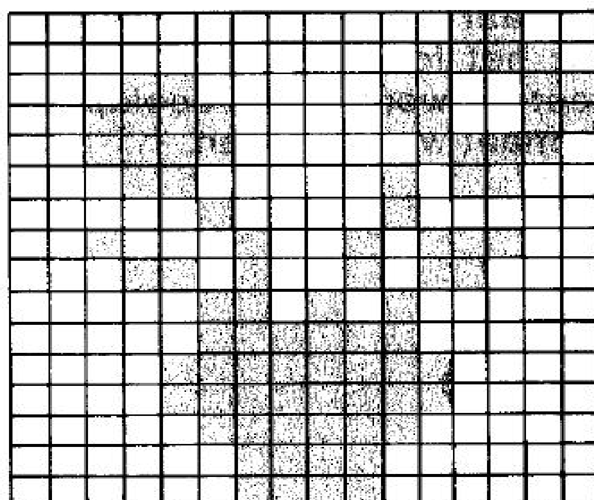
16×16 stippen,
standaard



×8 stippen,
ergroot



16×16
stippen,
vergroot



Bepalen van het beeldpatroonformaat—de SCREEN instruktie
De tweede parameter van de SCREEN instruktie dient voor het bepalen van het formaat van de beeldpatronen.

SCREEN schermsoort, formaat beeldpatronen

Parameter	Ingesteld formaat
0	8×8 stippen, standaard
1	8×8 stippen, vergroot
2	16×16 stippen, standaard
3	16×16 stippen, vergroot

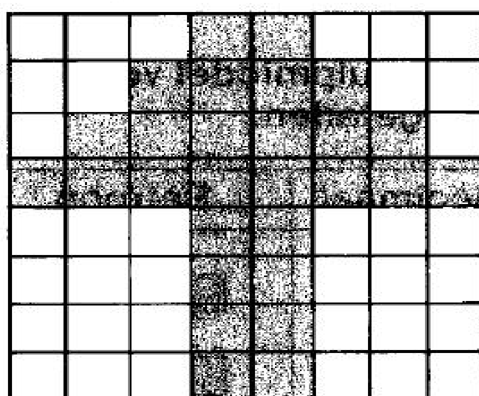
Als het bevel bijvoorbeeld

```
SCREEN 2,3
```

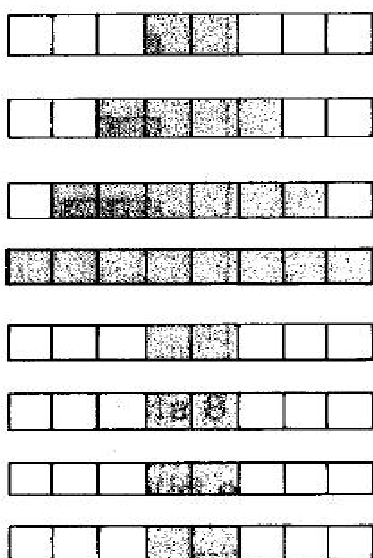
luidt, dan wordt de SCREEN 2 beeldschermsoort gekozen en voor het formaat van beeldpatronen het 16×16 stippen, vergroot, formaat. Nadat het formaat aldus met de SCREEN instruktie eenmaal is ingesteld, worden alle beeldpatronen op alle beeldvlakken in dit zelfde formaat weergegeven.

SAMENSTELLEN VAN EEN BEELDPATROON
SPRITE$ variabele

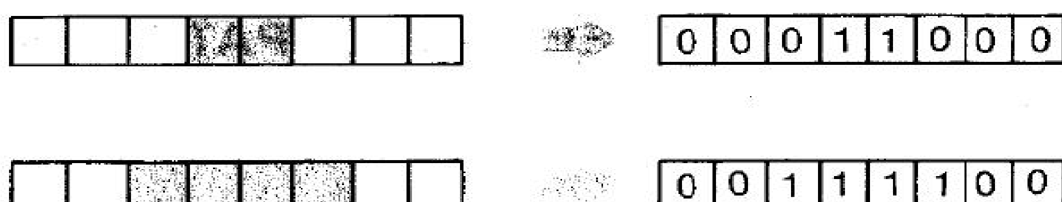
Het 8×8 stippen beeldpatroon
Voor het samenstellen van een beeldpatroon van 8×8 stippen moet het patroon eerst in 8 horizontale lagen opgesplitst worden. Een pijlvormig patroon wordt bijvoorbeeld samengesteld zoals in de onderstaande afbeelding.



Elk van de horizontale lagen vormt op zich dan een deelpatroon dat bestaat uit 8 beeldpunten.



Vervolgens wordt aan een "ingevulde" stip de waarde 1 toegekend en aan een "lege" stip een 0, zodat het deelpatroon uit te drukken valt als een binair getal. De bovenste horizontale laag wordt bijvoorbeeld 00011000 en de tweede laag 00111100.

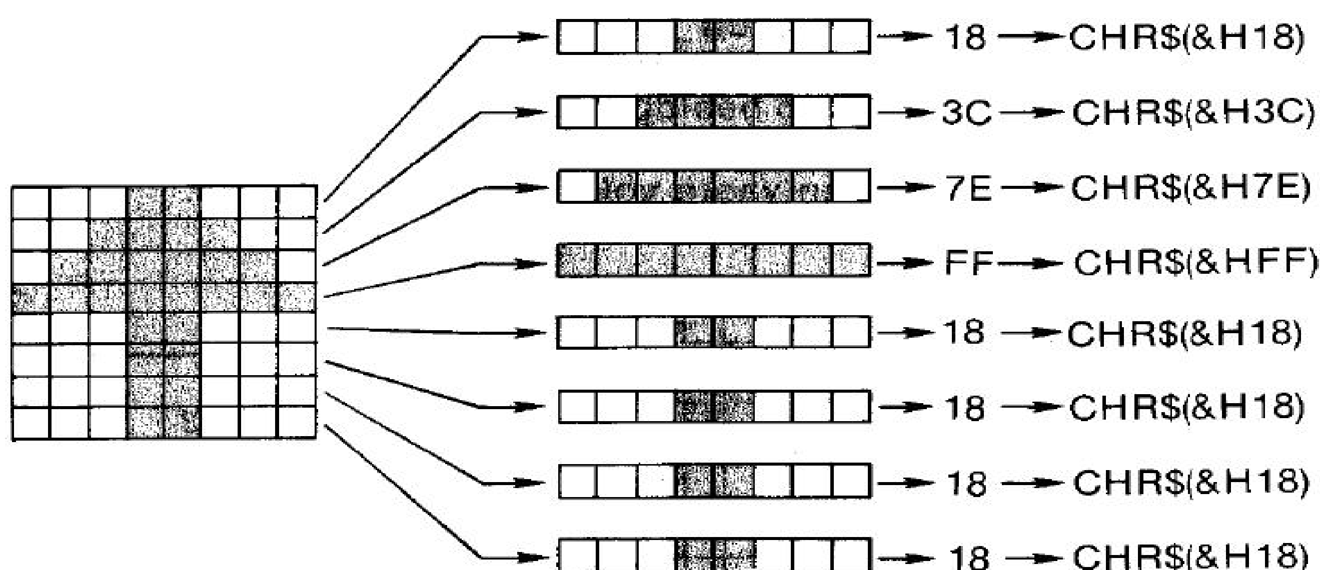


De binaire getallen worden daarna omgezet in hexadecimale of decimale (gewone) getallen. De bovenste laag wordt: 00011000 (binair) = 18 (hexadecimaal) of 24 (decimaal). De tweede laag wordt 00111100 (binair) = 3C (hexadecimaal) of 60 (decimaal).

De volgende tabel dient als hulpmiddel voor het omzetten van binaire getallen in hexadecimale getallen.

Patroon	Hexadecimaal	Patroon	Hexadecimaal
□□□□	0	■□□□	8
□□□■	1	■□□■	9
□□■□	2	■□■□	A
□□■■	3	■□■■	B
□■□□	4	■■□□	C
□■□■	5	■■□■	D
□■■□	6	■■■□	E
□■■■	7	■■■■	F

Eerst wordt het deelpatroon van 8 stippen nog verder opgesplitst, in de vier stippen links en de vier stippen aan de rechterkant. Vervolgens zijn in de tabel hierboven de bijbehorende hexadecimale cijfers (en letters) af te lezen.
Bij het patroon □□□■■□□□ zijn de linker vier stippen □□□■, hetgeen overeenkomt met het cijfer 1 hexadecimaal, zoals in de bovenstaande tabel af te lezen is; de rechter vier stippen zijn ■□□□, hetgeen volgens de tabel 8 is. Dit maakt het hexadecimale equivalent van het gehele patroon 18.
Tenslotte dienen we voor het invoeren te bepalen voor welk letterteken het hexadecimale (of decimale) getal de lettertekencode vormt. Dit doen we met behulp van de CHR$ funktie, zoals hieronder aangegeven.



De gegevens voor het 8×8 beeldpatroon, gevormd door de verkregen lettertekens, worden daarna van boven naar beneden opgeteld, d.i. samengevoegd, en in deze volgorde aan de SPRITE$ variabele toegewezen, waarmee de samenstelling van het gewenste beeldpatroon een feit is. Het pijlvormige patroon uit ons voorbeeld wordt als volgt samengesteld:

```
SPRITE$(1)=CHR$(&H18)+CHR$(&H3C)+CHR$(&H
7E)+CHR$(&HFF)+CHR$(&H18)+CHR$(&H18)+CHR
$(&H18)+CHR$(&H18)
```

Het samengestelde beeldpatroon krijgt het nummer 1, zoals aangegeven wordt door het nummer 1 tussen de haakjes van SPRITE$ (1). (Het gebruik van funkties wordt uitgelegd in hoofdstuk 7).

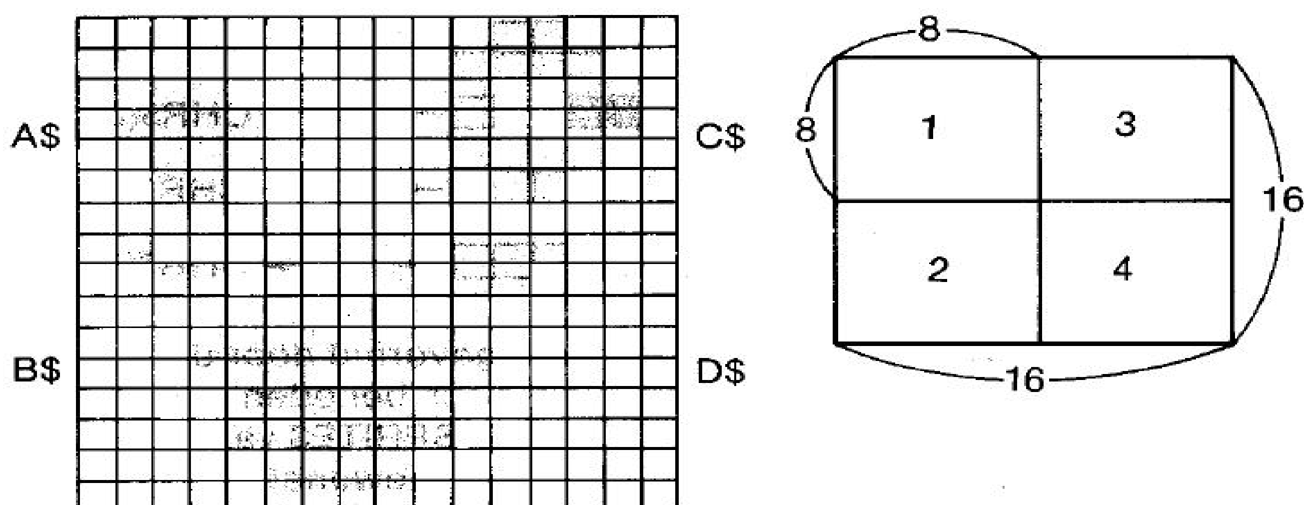
SPRITE$ (nummer beeldpatroon) = lettertekenrij

Als er bovendien voor de lettertekencode van de CHR$ funktie een letterteken bestaat dat gegenereerd kan worden, dan is het mogelijk dit laatste, in plaats van de CHR$ funktie en code, in de lettertekenrij op te nemen. In de bovenstaande programmaregel treffen we bijvoorbeeld CHR$ (&H3C) aan, wat staat voor het "<" teken, en CHR$ (&H7E), wat het "~" accent is. Deze twee tekens zijn dan ook rechtstreeks in de lettertekenrij op te nemen, als volgt:

```
SPRITE$(1)=CHR$(&H18)+"<"+"~"+CHR$(&HFF)
+CHR$(&H18)+CHR$(&H18)+CHR$(&H18)+CHR$(&
H18)
```

(Voor het omzetten van lettertekencodes in lettertekens kun je de codetabel in het MSX2-BASIC HANDBOEK VOOR HET PROGRAMMEREN raadplegen.)

Het 16×16 stippen beeldpatroon
Het samenstellen van een beeldpatroon van 16×16 stippen gaat op soortgelijke wijze. Een 16×16 beeldpatroon wordt echter beschouwd als een kwartet van vier 8×8 beeldpatronen. Deze vier kleinere patronen worden daarvoor in een vaste volgorde samengesteld:




```
A$=CHR$(&H0)+CHR$(&H0)+CHR$(&H18)+CHR$(&
H3C)+CHR$(&H3C)+CHR$(&H18)+CHR$(&H4)+CHR
$(&H22)
B$=CHR$(&H1A)+CHR$(&H6)+CHR$(&HF)+CHR$(&
HF)+CHR$(&H7)+CHR$(&H7)+CHR$(&H3)+CHR$(&
H3)
C$=CHR$(&HC)+CHR$(&H1E)+CHR$(&H33)+CHR$(
33)+CHR$(&H1E)+CHR$(&H2C)+CHR$(&H20)+CHR
$(&H5C)
D$=CHR$(&H58)+CHR$(&HA0)+CHR$(&HF0)+CHR$
(&HF0)+CHR$(&HE0)+CHR$(&HE0)+CHR$(&HC0)+
CHR$(&HC0)
SPRITE$(2)=A$+B$+C$+D$
```

Maximaal aantal tegelijk samen te stellen beeldpatronen
Het aantal tegelijk te gebruiken beeldpatronen van 8×8 stippen is 256, te nummeren van 0 t/m 255, en het maximale aantal beeldpatronen van 16×16 stippen is 64, genummerd van 0 t/m 63.

WEERGEVEN VAN EEN BEELDPATROON PUT SPRITE

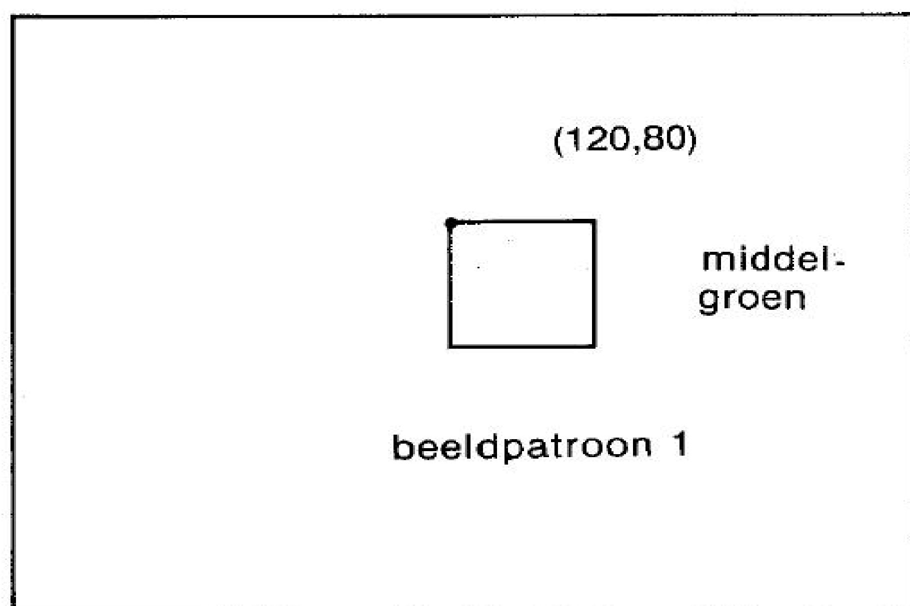
Het PUT SPRITE bevel dient voor het weergeven van een volledig samengesteld beeldpatroon (sprite) op één van de beeldvlakken.

PUT SPRITE beeldvlaknummer, [(X,Y)], [kleur], [nummer beeldpatroon]

Het PUT SPRITE bevel geeft het beeldpatroon met het gekozen nummer weer in de gekozen kleur op het gekozen beeldvlak.
Voor het weergeven van het eerder samengestelde beeldpatroon (patroonnummer 1) op beeldvlak 0 in middelgroen (kleurcode 2) op de plaats (120,80) geef je het bevel:

```
PUT SPRITE 0,(120,80),2,1
```

De getallen die voor de plaats van het beeldpatroon gekozen zijn geven het punt aan waar de linkerbovenhoek van het beeldpatroon op het scherm komt te staan. De X- en Y-coördinaat gelden binnen een coördinatenstelsel op het grafische scherm waarvan het punt van oorsprong (0,0) samenvalt met het punt dat gewoonlijk als (0, – 1) geldt.



Regels voor het weergeven van beeldpatronen (voor SCREEN 1, 2, 3)

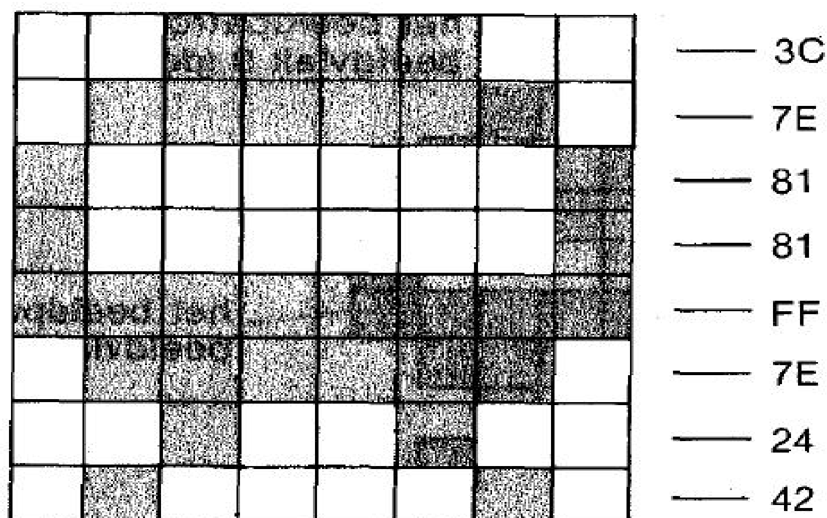
- Per beeldvlak kan slechts één beeldpatroon weergegeven worden.
- Wanneer meerdere beeldpatronen op verschillende beeldvlakken elkaar overlappen, dan verdwijnt het patroon op een achterliggend (hoger genummerd) beeldvlak achter het beeldpatroon dat meer naar voren ligt.
- Als zich op een en dezelfde horizontale lijn 5 of meer beeldpatronen bevinden, dan worden slechts de 4 beeldpatronen met voorrang, d.w.z. die op lager genummerde beeldvlakken, weergegeven.
- Bij weglaten van de plaats van een beeldpatroon uit het bevel worden automatisch voor de plaats de coördinaten uit het laatst verwerkte grafische bevel aangehouden.
- Bij weglaten van de kleurcode wordt voor een beeldpatroon de kleur van de voorgrond aangehouden.
- Bij weglaten van het beeldpatroonnummer wordt automatisch hiervoor het beeldvlaknummer aangehouden.

BEWEGEN VAN BEELDPATRONEN

Een beeldpatroon kan bewogen of verplaatst worden door herhaaldelijk een PUT SPRITE bevel uit te voeren, met telkens een nieuwe plaatsaanduiding in het bevel. Iedere keer dat een volgend PUT SPRITE bevel wordt uitgevoerd, verdwijnt automatisch het vorige beeldpatroon op hetzelfde beeldvlak, zodat het dus niet nodig is telkens weer een weergegeven beeldpatroon te wissen. Bovendien is het mogelijk de plaats van een beeldpatroon in stapjes van slechts een stip tegelijk te wijzigen, hetgeen de beweging van het beeldpatroon gelijkmatig en vloeiend maakt. Het volgende programma beweegt een beeldpatroon, dat er uitziet als een UFO, diagonaal over het scherm.

```
10 SCREEN 2,0
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 COLOR ,1,1:CLS
40 X=120:Y=50:VX=1:VY=1
50 PUT SPRITE 0,(X,Y),5,0
60 X=X+VX
70 IF X>240 OR X<0 THEN VX=-VX
80 Y=Y+VY
90 IF Y>180 OR Y<0 THEN VY=-VY
100 GOTO 50
```

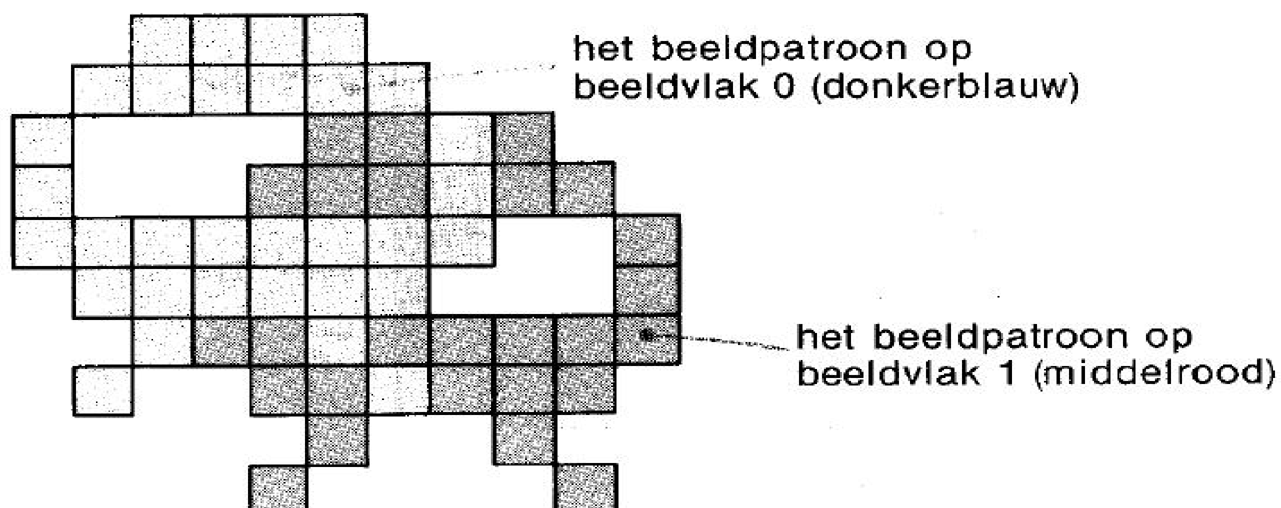
De beeldschermsoort is SCREEN 2 en het beeldpatroonformaat is 8×8 stippen, standaard formaat (regel 10).
De volgende afbeelding toont het beeldpatroon zoals dat samengesteld is in regel 20.



Dit beeldpatroon wordt weergegeven met behulp van het PUT SPRITE bevel in regel 50. De waarden van (X,Y), die de plaats van het beeldpatroon bepalen, worden eerst ingesteld op (120,50). Vervolgens worden deze waarden in de regels 60 t/m 80 gewijzigd, waarna het programma terugkeert naar regel 50. Hierdoor zie je het beeldpatroon over het scherm "vliegen".
Het volgende programma toont wat er gebeurt wanneer twee beeldpatronen elkaar overlappen.

```
10 SCREEN 2,1
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 COLOR ,1,1:CLS
40 FOR X=0 TO 117
50 PUT SPRITE 0,(X,80),4,0
60 PUT SPRITE 1,(240-X,84),8,0
70 NEXT X
80 GOTO 80
```

Om de overlapping zo duidelijk mogelijk te maken hebben we hier gebruik gemaakt van het 8×8 stippen vergroot formaat. Het beeldpatroon is verder hetzelfde als in het vorige programma. Dit patroon wordt weergegeven op beeldvlakken 0 en 1 door de twee PUT SPRITE bevelen in regel 50 en 60. Het beeldpatroon op beeldvlak 0 is donkerblauw en beweegt van links naar rechts, terwijl het patroon op beeldvlak 1 middelrood is en van rechts naar links beweegt. Wanneer de twee beeldpatronen elkaar overlappen, dan wordt het patroon op beeldvlak 0 (donkerblauw) over het andere beeldpatroon heen afgebeeld.



GEBRUIK VAN EXTRA BEELDPATROONFUNKTIES

- Uitgebreide beeldpatroonfunkties
- Beeldpatronen van kleur laten wisselen—COLOR SPRITE
- Beeldpatronen per lijn inkleuren—COLOR SPRITE$
- Methoden om beeldpatronen samen te stellen

UITGEBREIDE BEELDPATROONFUNKTIES

Beeldpatronen met extra funkties kunnen gebruikt worden bij de beeldschermsoorten SCREEN 4 t/m SCREEN 8.
De volgende tabel geeft aan in welke zin de funkties van beeldpatronen bij SCREEN 4 t/m 8 uitgebreid zijn in vergelijking met beeldschermsoorten SCREEN 1 t/m 3.

Funktie	SCREEN 1 t/m 3	SCREEN 4 t/m 8
Aantal beeldpatronen dat op een horizontale lijn weer te geven is	Maximaal 4	Maximaal 8
Kleuren beeldpatroon	1 kleur per beeldpatroon	8×8 stippen: maximaal 8 kleuren 16×16 stippen: maximaal 16 kleuren voor 1 beeldpatroon
Wijzigen van de kleur van een beeldpatroon	Kleur bepalen in het PUT SPRITE bevel	Kleur bepalen in het PUT SPRITE of COLOR SPRITE bevel

BEELDPATRONEN VAN KLEUR LATEN WISSELEN
COLOR SPRITE

De kleur van een beeldpatroon is te bepalen in het PUT SPRITE bevel, maar daarnaast kan de kleur van een beeldpatroon bij de beeldschermsoorten SCREEN 4 t/m 8 ook na afbeelden van het patroon nog gewijzigd worden met het COLOR SPRITE bevel.

COLOR SPRITE (nummer beeldvlak) = paletnummer

Het COLOR SPRITE bevel wijzigt de kleur van een beeldpatroon op het aangegeven beeldvlak in de gekozen kleur.

Om bijvoorbeeld de kleur van het beeldpatroon dat is afgebeeld op beeldvlak 2 te wijzigen in middelrood (kleurcode 8), geef je het volgende bevel:

```
COLOR SPRITE(2)=8
```

Bij het uitvoeren van het volgende programma landen er 6 UFO's van verschillende kleuren, waarbij deze na het landen van kleur wisselen. Na de laatste kleurwisseling zijn ze transparant, waarna de landingsprocedure weer herhaald wordt.

```
10 SCREEN 5,1
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 COLOR ,1,1:CLS
40 X1=0:Y1=118
50 FOR L=1 TO 14
60 READ X2,Y2                                  tekent de
70 LINE (X1,Y1)-(X2,Y2),3                      achtergrond
80 X1=X2:Y1=Y2
90 NEXT L
100 PAINT (1,119),3
110 P=0:X=5:YE=101:C=3:GOSUB 250
120 P=1:X=45:YE=122:C=4:GOSUB 250              beweegt de
130 P=2:X=83:YE=87:C=5:GOSUB 250               beeldpatronen
140 P=3:X=133:YE=143:C=6:GOSUB 250
150 P=4:X=168:YE=127:C=7:GOSUB 250
160 P=5:X=218:YE=96:C=8:GOSUB 250
170 FOR S=1 TO 10
180 FOR SP=0 TO 5
190 SC=S+3:IF S=10 THEN SC=0                   wijzigt de kleur van
200 COLOR SPRITE (SP)=SC                       de beeldpatronen
210 NEXT SP
220 FOR T=0 TO 300:NEXT T
230 NEXT S
240 GOTO 110
250 FOR Y=-8 TO YE                             subroutine voor
260 PUT SPRITE P,(X,Y),C,0                     het bewegen van
270 NEXT Y                                     de beeldpatronen
280 RETURN
290 DATA 26,118,40,139,67,139                  gegevens voor
300 DATA 79,104,103,104,128,160                het tekenen van
310 DATA 154,160,161,144,192,144               de achtergrond
320 DATA 200,113,252,113,252,212
330 DATA 0,212,0,118
```

Het COLOR SPRITE bevel in regel 200 wijzigt de kleur van de beeldpatronen. De beeldvlaknummers worden als waarden toegewezen aan de variabele SP. De variabele voor de kleurcode, SC, doorloopt de waarden van 3 t/m 13, maar bij de laatste herhaling van de lus (als S = 10), wordt de kleurcode 0 (transparant).

BEELDPATRONEN PER LAAG INKLEUREN COLOR SPRITE$

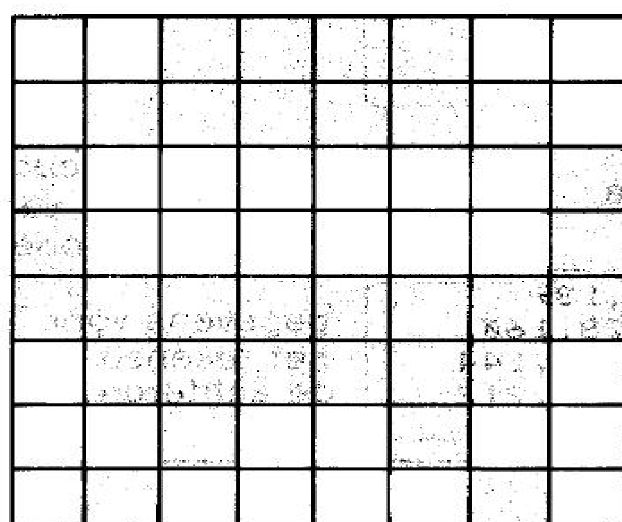
Een beeldpatroon van 8×8 stippen bestaat uit acht horizontale banden of lagen, terwijl een 16×16 stippen beeldpatroon 16 horizontale lagen beslaat. Bij de beeldschermsoorten SCREEN 4 t/m 8 bestaat de mogelijkheid de kleur van elk van deze lagen of banden afzonderlijk te bepalen.

COLOR SPRITE$ (beeldvlaknummer) = lettertekenrij

In het COLOR SPRITE$ bevel bepaalt een lettertekenrij de kleur van elk van de horizontale lagen van een beeldpatroon op het aangegeven beeldvlak.
De lettertekenrij in het COLOR SPRITE$ bevel is opgebouwd uit een aantal

CHR$ (kleurcode)

rijtjes samen, gescheiden door + tekens. De eerste CHR$ (kleurcode) bepaalt de kleur van de bovenste laag van het beeldpatroon, de tweede CHR$ (kleurcode) de kleur van de tweede laag, enzovoort.



—— laag 1 kleurcode ... 1
—— laag 2 kleurcode ... 2
—— laag 3 kleurcode ... 3
—— laag 4 kleurcode ... 4
—— laag 5 kleurcode ... 5
—— laag 6 kleurcode ... 6
—— laag 7 kleurcode ... 7
—— laag 8 kleurcode ... 8

Het volgende COLOR SPRITE$ bevel stelt de kleuren uit de bovenstaande afbeelding in voor een beeldpatroon op beeldvlak 0.

```
COLOR SPRITE$(0)=CHR$(1)+CHR$(2)+CHR$(3)
+CHR$(4)+CHR$(5)+CHR$(6)+CHR$(7)+CHR$(8)
```

Als in plaats van alle acht slechts zeven of minder CHR$ (kleurcode) rijtjes worden gegeven, dan blijft de kleur van de niet genoemde lagen ongewijzigd.
In het bevel

```
COLOR SPRITE(0)=CHR$(1)+CHR$(2)
```

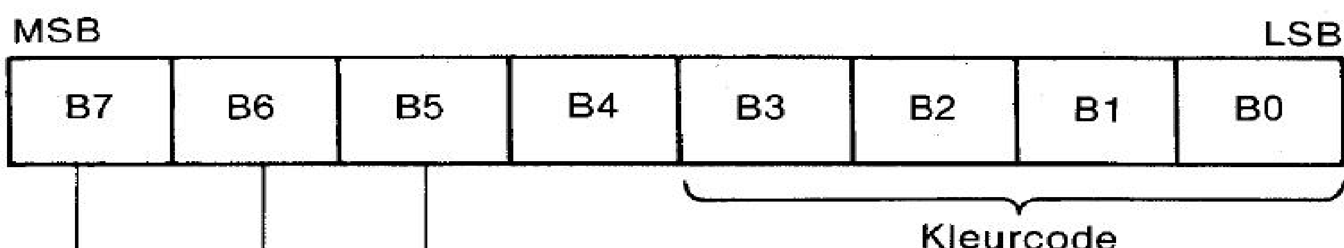
wordt de kleur van laag 1 gelijk aan kleurcode 1 en de kleur van laag 2 wordt kleurcode 2, maar de kleur van laag 3 en de lagen daaronder blijft ongewijzigd.
Het volgende programma is een gewijzigde versie van het vorige; het dient voor het weergeven van UFO's met een middelrood en donkerblauw strepenpatroon.

```
10 SCREEN 5,1
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 COLOR ,1,1:CLS
40 X1=0:Y1=118
50 FOR L=1 TO 14
60 READ X2,Y2
70 LINE (X1,Y1)-(X2,Y2),3
80 X1=X2:Y1=Y2
90 NEXT L
100 PAINT (1,119),3
110 P=0:X=5:YE=101:C=3:GOSUB 250
120 P=1:X=45:YE=122:C=4:GOSUB 250
130 P=2:X=83:YE=87:C=5:GOSUB 250
140 P=3:X=133:YE=143:C=6:GOSUB 250
150 P=4:X=168:YE=127:C=7:GOSUB 250
160 P=5:X=218:YE=96:C=8:GOSUB 250
170 FOR SP=0 TO 5
180 COLOR SPRITE$(SP)=CHR$(8)+CHR$(4)+CH
R$(8)+CHR$(4)+CHR$(8)+CHR$(4)+CHR$(8)+CH
R$(4)
190 NEXT SP
200 GOTO 200
250 FOR Y=-8 TO YE
260 PUT SPRITE P,(X,Y),C,0
270 NEXT Y
280 RETURN
290 DATA 26,118,40,139,67,139
300 DATA 79,104,103,104,128,160
310 DATA 154,160,161,144,192,144
320 DATA 200,113,252,113,252,212
330 DATA 0,212,0,118
```

Regel 180 in het gewijzigde deel van het programma bepaalt het rood/blauw strepenpatroon van de beeldpatronen.

Gegevens voor de lettertekenrijtjes

De gegevens die gebruikt kunnen worden voor de CHR$ funkties in het COLOR SPRITE$ bevel zijn niet beperkt tot enkel kleurcodes. Wanneer de gegevens in binaire notatie worden weergegeven, dan hebben de afzondelijke bits de volgende funkties:



Bij de waarde 1 worden overlappingen van beeldpatronen genegeerd.

Bij de waarde 1 worden overlappingen en voorrang van beeldpatronen genegeerd. Wanneer beeldpatronen elkaar overlappen, wordt een logische OR-bewerking met de kleurcodes uitgevoerd, en de uitkomst hiervan wordt dan de nieuwe kleurcode van het overlappende gebied.

Bij de waarde 1 wordt de laag 32 beeldpunten naar links verplaatst.

Het overlappen van beeldpatronen komt meer uitgebreid aan de orde in hoofdstuk 8.

Wanneer B7 de waarde 1 heeft en de kleurcode (B3—B0) is ingesteld op 0010 (= decimaal 2, middelgroen), dan wordt het volledige gegeven 10000010. En aangezien 10000010 in hexadecimale notatie &H82 is, wordt door het uitvoeren van

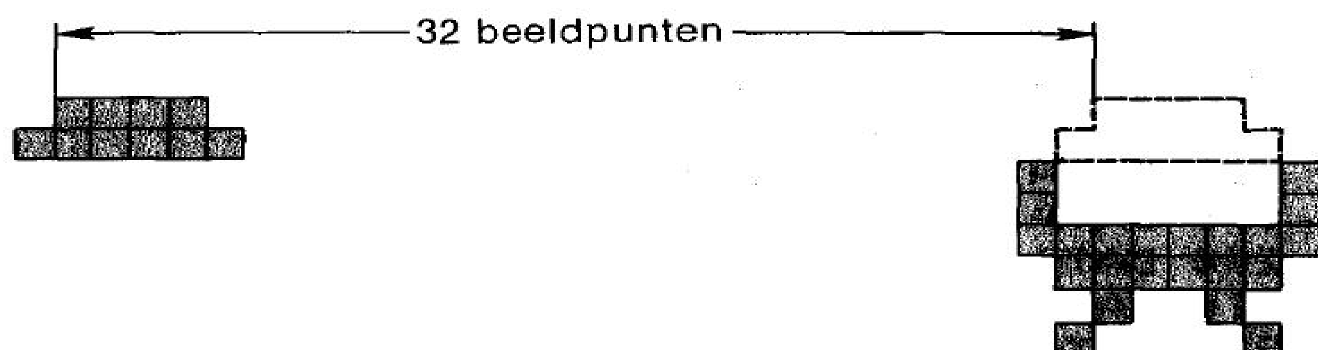
```
COLOR SPRITE$(0)=CHR$(&H82)
```

de kleur van de bovenste laag van het beeldpatroon op beeldvlak 1 gewijzigd in middelgroen, en wordt de laag 32 stippen naar links verschoven.

Bij verwerking van het volgende programma gaat het luik van de UFO open en komt een (buitenaards?) wezen tevoorschijn.

```
10 SCREEN 5,1
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 SPRITE$(1)=CHR$(&H58)+CHR$(&H58)+CHR$
(&H7E)+CHR$(&H1A)+CHR$(&H18)+CHR$(&H18)+
CHR$(&H0)+CHR$(&H0)
40 CLS
50 PUT SPRITE 0,(120,100),1,0
60 FOR T=0 TO 1000:NEXT T
70 COLOR SPRITE$(0)=CHR$(&H81)+CHR$(&H81
)
80 FOR T=0 TO 1000:NEXT T
90 PUT SPRITE 1,(120,96),14,1
100 GOTO 100
```

Regel 70 verschuift de eerste en tweede laag van het beeldpatroon (patroonnummer 0) dat is weergegeven op beeldvlak 0 naar links over een afstand van 32 beeldpunten.

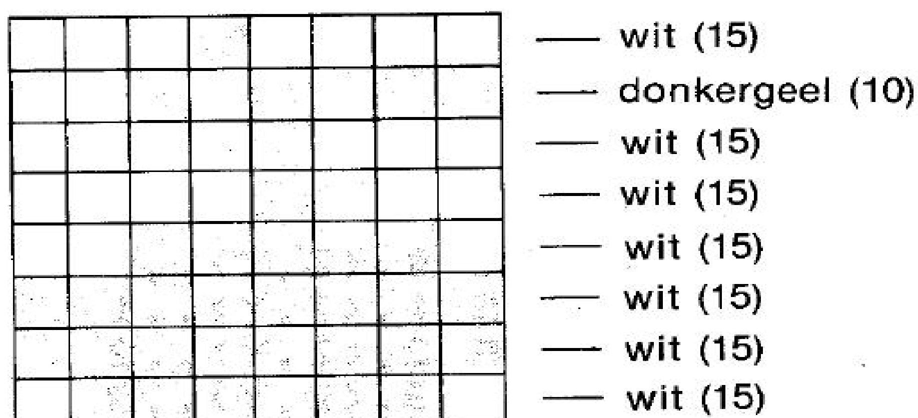


VISUELE METHODE OM BEELDPATRONEN SAMEN TE STELLEN

De visuele methode voor het samenstellen van een beeldpatroon bedient zich van de CHR$ funktie om de gegevens voor het patroon als waarde aan de SPRITE$ variabele toe te wijzen. De gegevens voor het beeldpatroon worden in de DATA instrukties geschreven in de vorm van gewone punten of stippen (.) en hoofdletters O, om zo de uiteindelijke vorm van het beeldpatroon al in het programma af te kunnen lezen.

```
10 SCREEN 5,1
20 SP$="":SC$=""
30 FOR SI=0 TO 7
40 READ SQ$,SC:SP=0
50 FOR SJ=1 TO 8
60 SP=SP*2-(MID$(SQ$,SJ,1)="O")
70 NEXT SJ
80 SP$=SP$+CHR$(SP):SC$=SC$+CHR$(SC)
90 NEXT SI
100 SPRITE$(0)=SP$
110 COLOR SPRITE$(0)=SC$
120 PUT SPRITE 0,(120,90),,0
130 GOTO 130
140 '
150 DATA ...O....,15
160 DATA ..O.OOOO,10
170 DATA ...OO...,15
180 DATA ....OO..,15
190 DATA ..OOOOO.,15
200 DATA OOOOOOOO,15
210 DATA OOOOOOOO,15
220 DATA .OOOOOO.,15
```

In de DATA instrukties van de regels 150 t/m 220 wordt het samen te stellen beeldpatroon uitgedrukt in stippen en O's. Een stip staat voor een "leeg" beeldpunt en een O voor een "ingevuld" beeldpunt. Het getal aan het eind van elke DATA instruktie is de kleurcode die de kleur van de betreffende laag van het beeldpatroon bepaalt. De regels 20 t/m 110 dienen voor het toewijzen van de gegevens in de DATA instrukties aan de SPRITE$ variabele, en het bepalen van de kleuren met het COLOR SPRITE$ bevel. In regel 60 wordt een funktie gebruikt (die wordt uitgelegd in hoofdstuk 7) om de gegevens in hexadecimale getallen om te zetten. De regels 100 en 110 stellen het beeldpatroon samen en bepalen de kleuren. Zo wordt het volgende beeldpatroon samengesteld.



De manier waarop in dit programma het beeldpatroon wordt samengesteld zul je misschien een nogal geavanceerde programmeermethode vinden, maar als je het beschouwt als een formule voor het vormen van een patroon zul je al gauw zien dat het in feite niet zo moeilijk is. Deze methode maakt het programma wel wat langer, maar het voordeel is dat al in de regels van het programma de vorm van het beeldpatroon duidelijk zichtbaar is. Je kan het dus makkelijk kontroleren en je hoeft voor het schrijven van het programma niet zelf elk gegeven in een hexadecimaal getal om te zetten. Bovendien is de vorm van het beeldpatroon uiterst gemakkelijk aan te passen, eenvoudigweg door een punt in een O te veranderen of een O in een punt.

Programmeervoorbeeld

In het onderstaande programma wordt voor het samenstellen van beeldpatronen de methode met de DATA instrukties gebruikt. Ook bevat het programma twee bevelen (DEFINT en DEFFN) die in dit boek niet nader uitgelegd worden. Zie voor een volledige beschrijving van deze bevelen het MSX2-BASIC HANDBOEK VOOR HET PROGRAMMEREN.

Het programma tekent een moedereend met kleine eendjes, zwemmend in een vijver.

```
10 SCREEN 5,1
20 DEFINT A-Z:X=0:Y=0:Z=0
30 DEFFNX=(Z+240)MOD 256
40 ' *** beeldpatronen ***
50 RESTORE 380:SN=0:GOSUB 260
60 RESTORE 470:SN=1:GOSUB 260
70 RESTORE 470:SN=2:GOSUB 260
80 RESTORE 470:SN=3:GOSUB 260
90 RESTORE 470:SN=4:GOSUB 260
100 RESTORE 470:SN=5:GOSUB 260
110 ' *** vijver tekenen ***
120 LINE (0,116)-(255,116),7
130 PAINT (0,118),7
140 ' *** patronen bewegen
150 Y=100
160 FOR X=0 TO 255
170  Z=X   :PUT SPRITE 0,(Z,Y),,0
180  Z=FNX:PUT SPRITE 1,(Z,Y),,1
190  Z=FNX:PUT SPRITE 2,(Z,Y),,2
200  Z=FNX:PUT SPRITE 3,(Z,Y),,3
210  Z=FNX:PUT SPRITE 4,(Z,Y),,4
220  Z=FNX:PUT SPRITE 5,(Z,Y),,5
230 NEXT X
240 GOTO 160
250 ' *** beeldpatronen subroutine ***
260 SP$="":SC$=""
270 FOR SI=0 TO 7
280 READ SQ$,SC:SP=0
290  FOR SJ=1 TO 8
300   SP=SP*2-(MID$(SQ$,SJ,1)="O")
310  NEXT SJ
320  SP$=SP$+CHR$(SP):SC$=SC$+CHR$(SC)
330 NEXT SI
340 SPRITE$(SN)=SP$
350 COLOR SPRITE$(SN)=SC$
360 RETURN
```

```
370 ' *** gegevens beeldpatroon ***
380 DATA ...O....,15
390 DATA ..O.OOOO,10
400 DATA ...OO...,15
410 DATA ....OO..,15
420 DATA ..OOOOO.,15
430 DATA OOOOOOOO,15
440 DATA OOOOOOOO,15
450 DATA .OOOOOO.,15
460 ' *** gegevens beeldpatroon ***
470 DATA ........,0
480 DATA ........,0
490 DATA ....O...,10
500 DATA ...O.OOO,8
510 DATA ....OO..,10
520 DATA .OO..OO.,10
530 DATA .OOOOOO.,10
540 DATA ..OOOO..,10
```

Hoofdstuk 7
Gebruik van funkties

NUMERIEKE FUNKTIES

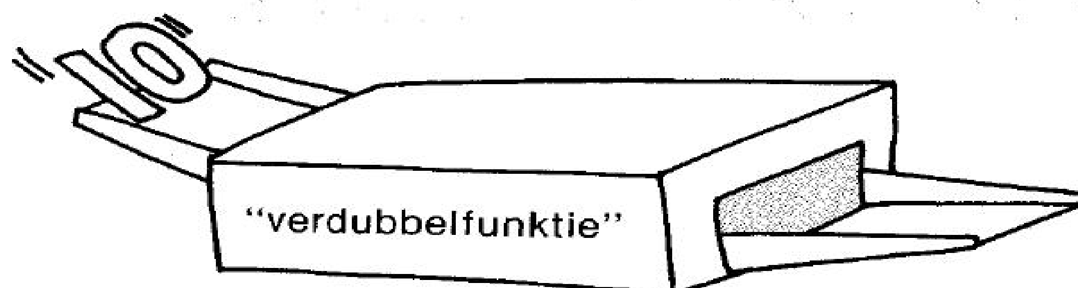
- Wat zijn funkties?
- Een paar getalsfunkties
- Funktie voor willekeurige getallen

WAT ZIJN FUNKTIES?

De woorden van de BASIC taal omvatten bevelen en funkties. Tot op heden hebben we ons in dit boek voornamelijk bezig gehouden met het gebruik van bevelen. Nu willen we echter het gebruik van funkties onder de loep nemen.

Een BASIC funktie kun je beschouwen als een machientje dat de waarde die je er in stopt verwerkt en vervolgens het resultaat geeft.

Neem bijvoorbeeld aan dat we een funktie hebben die een getal verdubbelt. (Zo'n funktie bestaat in werkelijkheid niet in MSX2-BASIC). Als je hierin het getal 10 zou stoppen, zou er 20 uit komen. Als je er 100 in stopt komt er 200 uit, 450 wordt 900 enzovoort. Welk getal je ook in de funktie stopt, het wordt telkens verdubbeld teruggegeven.



Als je er 10 in stopt



komt er 20 uit.

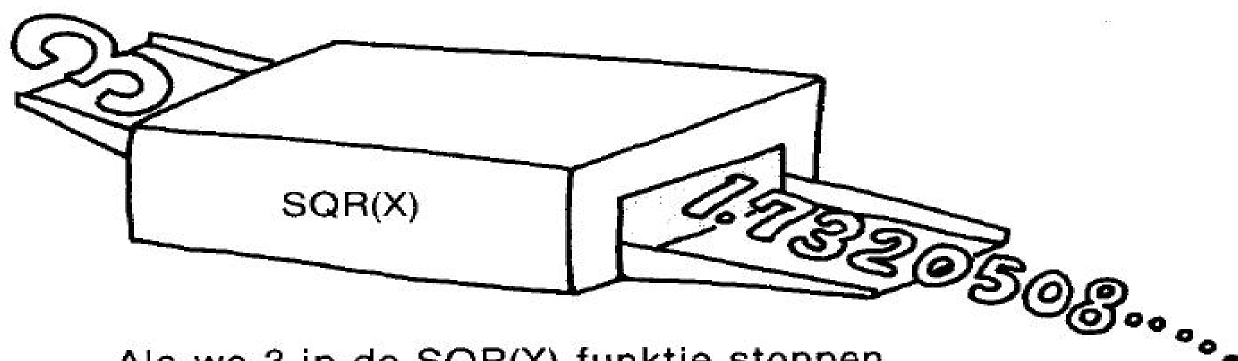
GETALSFUNKTIES

In MSX2-BASIC noemen we de funkties waarvan zowel de ingevoerde als de gegeven waarde getallen zijn numerieke of getalsfunkties. In totaal zijn er in MSX2-BASIC 17 van dergelijke getalsfunkties.

ABS(X)	CDBL(X)	CINT(X)	CSNG(X)
FIX(X)	INT(X)	SQR(X)	SGN(X)
ATN(X)	COS(X)	SIN(X)	TAN(X)
LOG(X)	EXP(X)	RND(X)	
ERR	ERL		

DE VIERKANTSWORTELFUNKTIE SQR(X)

Om te zien hoe de funkties werken nemen we als voorbeeld de wortelfunktie SQR(X). Deze funktie geeft als waarde de vierkantswortel van het getal X. Deze gegeven waarde kunnen we ook de "uitkomst" van de funktie noemen. Zo kunnen we dus zeggen: de SQR(X) funktie geeft als uitkomst de vierkantswortel van X.
Als X bijvoorbeeld gelijk is aan 3 dan wordt de wortel 1,7320508.... Als X gelijk is aan 5 dan wordt de wortel 2,2360679... .



Als we 3 in de SQR(X) funktie stoppen

Gebruik van funkties
Als de invoer van de SQR(X) funktie 3 is, dan wordt de uitkomst 1,7320508... . Om een getal in een funktie te stoppen voeren we deze als waarde voor X in.

```
SQR(3)
```

Nu is in de SQR(X) funktie 3 ingevoerd. Naast van deze direkte methode kunnen we ook eerst 3 als waarde aan een variabele toewijzen, en dan deze variabele in de funktie invoeren in plaats van het getal.
Als je eerst stelt dat

```
A=3
```

en dan

```
SQR(A)
```

schrijft, dan wordt het getal 3 in de SQR(X) funktie ingevoerd.
In beide gevallen, of je het getal direkt invoert of gebruik maakt van een variabele, de uitkomst van de funktie wordt 1,7320508... . Telkens wanneer je deze funktie in het programma gebruikt, wordt als uitkomst 1,7320508... gegeven.

De werkwijze voor het verkrijgen van de uitkomst van een funktie in een programma is als volgt. Aangezien de funkties op zichzelf geen bevelen zijn, kunnen ze niet zonder meer gebruikt worden om de computer instrukties te geven. Om de uitkomst van een funktie te verkrijgen dient de laatste in kombinatie met een bevel gebruikt te worden. Hiervoor kunnen we het LET bevel en het PRINT bevel gebruiken.

```
R=SQR(3)
```

Dit LET bevel wijst de uitkomst van SQR(3), d.i. 1,7320508... als waarde aan de getalsvariabele R toe. We kontroleren of dit naar behoren gebeurt.

```
R=SQR(3)
Ok
PRINT R
 1.7320508075688
Ok
```

De waarde van de vierkantswortel van 3, tot in dertien decimalen (dus in totaal 14 cijfers), is aan de variabele R toegewezen.
In MSX2-BASIC wordt de uitkomst van een berekening gewoonlijk gegeven tot op 14 cijfers nauwkeurig. Het is echter ook mogelijk de uitkomst tot op 6 cijfers te doen geven. Tot op veertien cijfers nauwkeurig wordt dubbele precisie genoemd, tot op 6 cijfers nauwkeurig enkele precisie.

De uitkomst van een funktie kan direkt op het scherm worden afgebeeld, eenvoudigweg met een PRINT bevel.

```
PRINT SQR(5)
 2.2360679774998
Ok
```

Funkties zijn ook te gebruiken binnen een IF—THEN voorwaardelijke uitdrukking.

```
IF SQR(A)>=10 THEN END
```

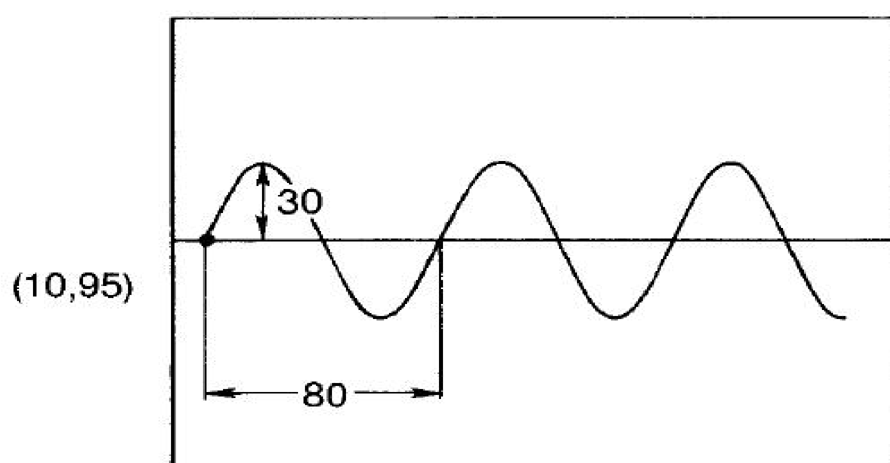
De uitkomst van SQR(A) is afhankelijk van de waarde van A. Het hier gebruikte IF—THEN bevel houdt in dat het programma ten einde is wanneer de uitkomst van SQR(A) de 10 overschrijdt.
Zoals we eerder zagen verwerkt een funktie een ingevoerde waarde, volgens vaste regels die voor elke funktie precies zijn gedefinieerd, en geeft vervolgens de uitkomst. De gegeven waarde van de uitkomst kan gebruikt worden in kombinatie met een aantal van de BASIC bevelen zoals LET, PRINT en IF—THEN.

TRIGONOMETRISCHE FUNKTIES SIN(X)

De trigonometrische funktie SIN(X) is één van de numerieke of getalsfunkties. Deze funktie geeft als uitkomst de sinus van X. Andere trigonometrische funkties die deel uitmaken van MSX2-BASIC zijn COS(X) (cosinus), TAN(X) (tangens) en ATN(X) (boogtangens).
Laten we de SIN(X) funktie maar eens gebruiken voor een programma dat een sinuskromme tekent.

```
10 SCREEN 2:CLS
20 PI=3.14
30 LINE (0,95)-(252,95)
40 FOR X=10 TO 230
50 R=(X-10)*PI/40
60 S=SIN(R)
70 VY=S*30
80 Y=95-VY
90 PSET (X,Y)
100 NEXT X
110 GOTO 110
```

Voor toepassing in trigonometrische funkties als SIN(X), COS(X) en TAN(X) wordt de waarde van X uitgedrukt in radialen (180° = 2π radialen). In dit programma doorloopt X de waarden van 10 tot 230. Deze waarden worden in regel 50 omgezet in eenheden van radialen. Als de waarde van X 90 is dan wordt de waarde van R gelijk aan 2π radialen, als X 50 is, wordt R π radialen, en bij X = 10 is R gelijk aan 0. (X is hier de X-coördinaat van de punten op het scherm bij het tekenen van een sinuskromme.)
Voor al deze waarden wordt de uitkomst van de SIN(X) funktie in regel 60 als waarde aan de variabele S toegewezen. Wanneer de sinuswaarde 1 is, dan wordt de vertikale afstand van de kromme vanaf de X-as, d.w.z. vanaf de punten met Y = 0, gelijk aan 30 (regel 70). In regel 80 wordt de Y-coördinaat van de sinuskromme op het scherm bepaald door de Y-coördinaat van de oorsprong te kiezen als 95 op het scherm.
Bij verwerking van dit programma wordt op het scherm de volgende sinuskromme getekend.



DE ABSOLUTE-WAARDE FUNKTIE ABS(X)

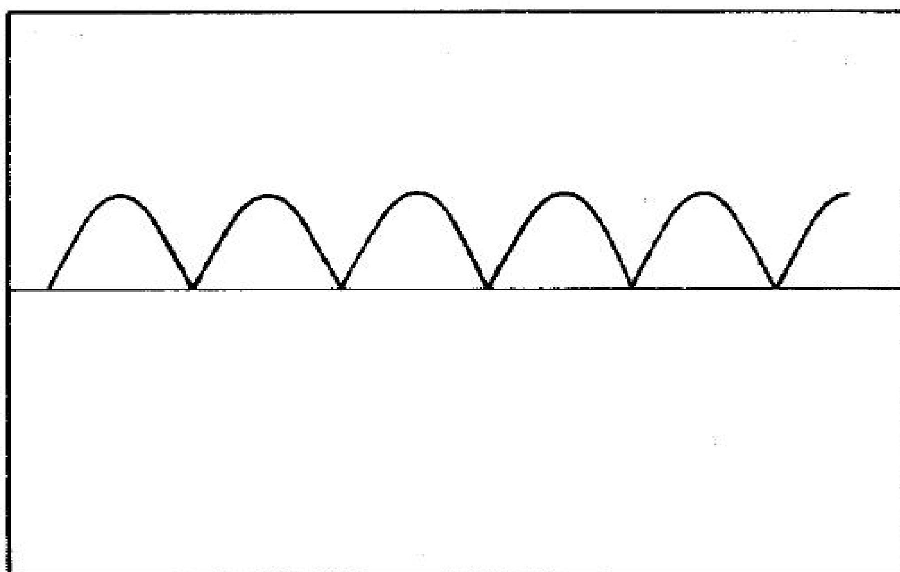
De ABS(X) funktie geeft als uitkomst de absolute waarde van X. Wanneer X bijvoorbeeld 100 is, dan geeft ABS(X) als uitkomst 100. Wanneer X gelijk is aan −100, dan geeft ook in dit geval ABS(X) de uitkomst 100.

```
PRINT ABS(-3.5)
 3.5
Ok
PRINT ABS(-10)+ABS(8)
 18
Ok
```

We kunnen de ABS(X) funktie toepassen op de sinuskromme van het laatste programma.

```
10 SCREEN 2:CLS
20 PI=3.14
30 LINE (0,95)-(252,95)
40 FOR X=10 TO 230
50 R=(X-10)*PI/40
60 S=SIN(R)
70 VY=ABS(S*30)
80 Y=95-VY
90 PSET (X,Y)
100 NEXT X
110 GOTO 110
```

Het programma is zo herzien dat in regel 70 aan de variabele VY de absolute waarde van SIN(R) * 30 wordt toegewezen. Als resultaat verschijnt de volgende afbeelding op het scherm.



DE WILLEKEURIG-GETAL FUNKTIE RND(X)

De RND(X) willekeurig-getal funktie **geeft een willekeurig getal tussen 0 en 1 als uitkomst** wanneer X een positief getal is, groter dan 0. Voer enkele maken rechtstreeks

```
PRINT RND(1)
```

in, als losstaand bevel.

```
PRINT RND(1)
 .59521943994623
OK
PRINT RND(1)
 .10658628050158
OK
PRINT RND(1)
 .76597651772823
```

Telkens wanneer de RND(X) funktie uitgevoerd wordt, geeft deze een willekeurig getal tussen 0 en 1 als uitkomst. Je kunt de RND(X) funktie gebruiken om willekeurige getallen binnen een bepaald bereik te produceren.

RND(X) EN DE GEHEEL-GETAL FUNKTIE INT(X)

Laten we eens nagaan hoe de RND(X) funktie te gebruiken is om willekeurige gehele getallen van 0 tot 200 als uitkomst te krijgen. RND(1) zelf zal alleen maar uitkomsten tussen 0 en 1 (van 0 tot 0,99999999999999) opleveren. Als we echter RND(1) met 200 vermenigvuldigen

```
RND(1)*200
```

dan zal de uitkomst een getal tussen 0 en 200 zijn. De uitkomst zal een getal zijn dat uit veertien cijfers bestaat (inclusief de cijfers achter de komma). De numerieke INT(X) funktie wordt gebruikt om deze uitkomst in een geheel getal te veranderen.
INT(X) verandert iedere waarde van X in een geheel getal en geeft dit gehele getal als uitkomst.

```
PRINT INT (12.73)
 12
Ok
PRINT INT (-7.99)
-8
Ok
```

INT(X) geeft als uitkomst het grootst mogelijke gehele getal dat lager is dan de waarde van X.
Vandaar dat als RND(1) * 200 de waarde van INT(X) is

```
INT(RND(1)*200)
```

de uitkomst een willekeurig getal tussen 0 en 200 (van 0 tot 199) zal zijn. Om als uitkomst een willekeurig getal tussen 0 en 200 te krijgen, hoef je dus alleen maar

```
INT((RND(1)*(200+1))
```

of

```
INT (RND(1)*201)
```

te gebruiken.

```
10 FOR L=1 TO 10
20 X=INT(RND(1)*201)
30 PRINT X;
40 NEXT L
```

Dit programma zal 10 willekeurige getallen tussen 0 en 200 genereren en weergeven, zoals in de onderstaande afbeelding getoond wordt.

```
RUN
 119  21  153  116  147  37  74  190
 128  94
```

De standaard schrijfwijze om een willekeurig getal tussen 0 en N te genereren, is:

X = INT(RND(1) * (N + 1))

Bij deze schrijfwijze wordt een willekeurig getal tussen 0 en N aan X toegewezen.

Voor het genereren van een willekeurig getal tussen M en N wordt de volgende schrijfwijze gebruikt:

X = INT(RND(1) * (N – M + 1)) + M

Om bijvoorbeeld een willekeurig getal tussen 2 en 15 te genereren, zou je voor M 2 en voor N 15 invullen. Vandaar dat N – M + 1 = 14.

X = INT(RND(1) * 14) + 2

Willekeurige vierkanten
Laten we een programma schrijven dat 50 vierkanten met verschillende afmetingen tekent en deze in verschillende kleuren op verschillende plaatsen op het scherm weergeeft.

```
10 SCREEN 5
20 COLOR ,1,1:CLS
30 FOR B=1 TO 50
40 SX=INT(RND(1)*220)+5
50 SY=INT(RND(1)*180)+5
60 ST=INT(RND(1)*40)+20
70 C=INT(RND(1)*14)+2
80 LINE (SX,SY)-STEP(ST,ST),C,BF
90 NEXT B
100 GOTO 100
```

De variabelen SX,SY zijn de linker bovencoördinaten van ieder vierkant, ST is de lengte van een zijde en C is de kleurcode. De regels 40 t/m 70 wijzen met behulp van de RND(X) funktie willekeurige getallen aan deze variabelen toe. Het cijferbereik van ieder van de willekeurige getallen is:

SX...5 tot 224
SY...5 tot 184
ST...20 tot 59
C...2 tot 15

Vervolgens tekent regel 80 vierkanten met verschillende afmetingen in verschillende kleuren op verschillende plaatsen op het scherm.

Telkens bij uitvoering van het programma verschillende willekeurige getallen genereren

Bij het verscheidene keren uitvoeren van het "Willekeurige vierkanten" programma zul je zien dat de vierkanten iedere keer op dezelfde plaats op het scherm worden gegeven. In de computer bestaat in feite een lijst met willekeurige getallen. Iedere keer wanneer er een programma dat gebruik maakt van de RND(1) funktie wordt uitgevoerd, geeft de RND(1) funktie een gedeelte van deze lijst als uitkomst. De uitkomsten worden in dezelfde volgorde gegeven, waarbij begonnen wordt met het eerste getal op de lijst.

Het is echter mogelijk een programma te maken waarbij er telkens bij uitvoering een ander gedeelte van de lijst met willekeurige getallen als uitkomst wordt gegeven. Hierdoor is bij het herhaald uitvoeren van het programma het resultaat iedere keer verschillend.

Een van de manieren om dit te doen is door gebruik te maken van het interne klokcircuit van de computer. Hiervoor moeten de volgende regels aan het begin van een programma worden toegevoegd.

```
22 FOR N=0 TO TIME-INT(TIME/100)*100
24 X=RND(1)
26 NEXT N
```

De in regel 22 gebruikte TIME variabele is een speciale MSX-BASIC variabele. De huidige waarde van het klokcircuit wordt altijd aan deze variabele toegewezen. De waarde van het interne klokcircuit verandert iedere 1/50 sekonde van 0 tot 65535 in eenheden van 1. Wanneer de waarde het getal 65535 heeft bereikt, keert deze terug naar 0 waarna het proces herhaald wordt.
Bij uitvoering van een programma waarin regels, zoals eerder getoond werden, zijn opgenomen zal de waarde van de TIME variabele een getal tussen 0 en 65535 zijn. Regel 22 bevat echter twee TIME variabelen. Gezien het feit dat de waarde van het interne klokcircuit in een relatief snel tempo verandert, zullen de waarden van beide TIME variabelen enigzins van elkaar verschillen. Wanneer er bijvoorbeeld een interne klokcircuitwaarde van 42280 als waarde aan de eerste TIME variabele is toegewezen, dan zal er een waarde van ongeveer 42281 aan de tweede TIME variabele toegewezen worden.
Daarom zou regel 22

```
FOR N=0 TO 42280-INT(42281/100)*100
```

of

```
FOR N=0 TO 80
```

zijn.
Of wanneer de waarde die aan de eerste TIME variabele toegewezen wordt, 10900 is, dan zal de waarde die aan de tweede TIME variabele toegewezen wordt, 10901 zijn. Het bevel zou er dan als volgt uitzien:

```
FOR N=0 TO 10900-INT(10901/100)*100
```

of

```
FOR N=0 TO 0
```

Op deze manier zal bij uitvoering van het programma de uiteindelijke waarde van N in het FOR—NEXT bevel veranderen, afhankelijk van de waarde van het interne klokcircuit. Wanneer de uiteindelijke waarde van N 80 is, dan zal X=RND(1) in regel 24 door de FOR—NEXT lus 81 keer uitgevoerd worden. Als gevolg hiervan zal de eerstvolgende keer dat de RND(1) funktie in het programma gebruikt wordt, het 82ste getal op de computerlijst met willekeurige getallen als uitkomst gegeven worden.
Laten we de genoemde drie regels aan het voorgaande "Willekeurige vierkanten" programma toevoegen:

```
10 SCREEN 5
20 COLOR ,1,1:CLS
22 FOR N=0 TO TIME-INT(TIME/100)*100
24 X=RND(1)
26 NEXT N
30 FOR B=1 TO 50
40 SX=INT(RND(1)*220)+5
50 SY=INT(RND(1)*180)+5
60 ST=INT(RND(1)*40)+20
70 C=INT(RND(1)*14)+2
80 LINE (SX,SY)-STEP(ST,ST),C,BF
90 NEXT B
100 GOTO 100
```

Er bestaat nog een methode om bij herhaald uitvoeren van hetzelfde programma iedere keer verschillende willekeurige getallen te verkrijgen. Bij deze methode wordt er gebruik gemaakt van de INKEY$ funktie, waarbij gegevens via het toetsenbord ingevoerd worden. Om gebruik te kunnen maken van de INKEY$ funktie moet je volgende drie regels aan het programma toevoegen:

```
22 A$=INKEY$
24 X=RND(1)
26 IF A$="" THEN 22
```

De INKEY$ funktie wordt op blz. 207 nader uitgelegd. Door het toevoegen van de eerdergenoemde drie regels, blijft RND(1) uitgevoerd worden, totdat er op een toets van het toetsenbord wordt gedrukt. Voeg deze drie regels aan het "Willekeurige vierkanten" programma toe en bekijk de resultaten.

TEKSTFUNKTIES

- Wat zijn tekstfunkties?
- Lettertekenrijtjes verwerken

WAT ZIJN TEKSTFUNKTIES?

Een tekstfunktie voert een bewerking met een lettertekenrij uit en geeft de uitkomst van deze bewerking weer als een rijtje lettertekens.

Er zijn zeven tekstfunkties bij MSX2-BASIC.

LEFT$(X$,N), MID$(X$,M[,N]), RIGHT$(X$,N)
SPACE$(N), STRING$(N,J) of STRING$(N,X$),
TAB(N), SPC(N)

Ondanks het feit dat er bij verschillende van deze funkties sprake is van het invoeren van numerieke waarden, waarbij de uitkomst een lettertekenrijtje is, vallen deze funkties toch onder de tekstfunkties.

INVOEGEN VAN SPATIES SPACE$(N), SPC(N)

De SPACE$(N) en de SPC(N) funktie geven als uitkomst een N aantal spaties. Het gebruik van een van beide funkties resulteert in hetzelfde, maar de SPC(N) funktie kan alleen in kombinatie met het PRINT bevel gebruikt worden.

Laten we eens bekijken welke resultaten het oplevert, als we beide funkties buiten een programma om gebruiken.

```
PRINT "A";SPACE$(10);"B"
A          B
Ok                          10 spaties
PRINT "C";SPC(15);"D"
C               D
Ok                          15 spaties
S$=SPACE$(5)
Ok
PRINT "X";S$;"Y";S$;"Z"
X     Y     Z
Ok                          5 spaties
```

Een tekstfunktie geeft altijd lettertekens als uitkomst. Daarom moet, als de waarde met een LET bevel toegewezen wordt, de variabele eveneens een rij-variabele zijn.
In het bovenstaande voorbeeld

```
S$=SPACE$(5)
```

worden er vijf spaties aan de S$ rij-variabele toegewezen.

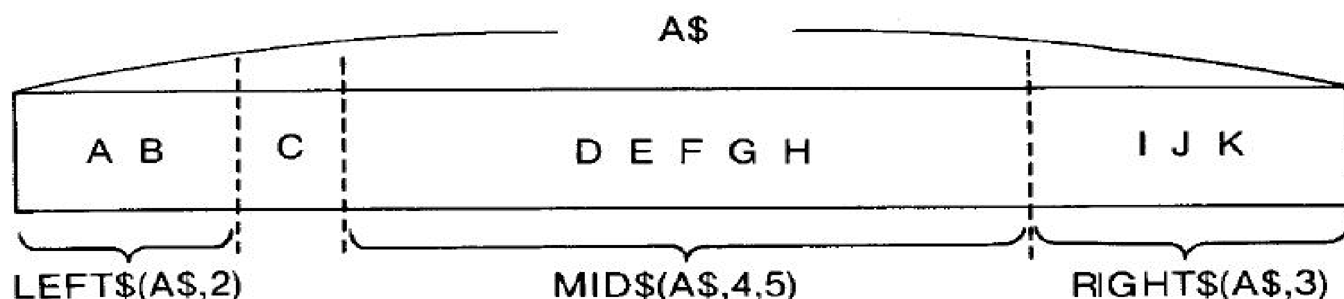
LETTERTEKENRIJTJES VERWERKEN
LEFT$(X$,N), MID$(X$, M, N), RIGHT$(X$, N)

De LEFT$(X$, N), MID$(X$, M, N), en RIGHT$(X$, N) tekstfunkties geven als uitkomst een deel van een bepaalde lettertekenrij.
LEFT$(X$,N) geeft een N aantal lettertekens vanaf de linkerkant van de lettertekenrij als uitkomst.
RIGHT$(X$,N) geeft een N aantal lettertekens vanaf de rechterkant van de lettertekenrij als uitkomst.
MID$(X$,M,N) geeft een N aantal lettertekens, waarbij begonnen wordt met het M-de letterteken in de rij.

De volgende losstaande PRINT bevelen laten de werking van deze drie funkties zien.

```
A$="ABCDEFGHIJK"
Ok
PRINT LEFT$(A$,2)
AB
Ok
PRINT RIGHT$(A$,3)
IJK
Ok
PRINT MID$(A$,4,5)
DEFGH
Ok
```

Zoals uit het bovenstaande blijkt, geven de LEFT$(X$,N), MID$(X$,M,N), en RIGHT$(X$,N) funkties als uitkomst een bepaald gedeelte van een lettertekenrij.



Bij uitvoering van het volgende programma worden alle lettertekenrij-en (namen van personen) die ingevoerd worden met het ENTER bevel weergegeven. Vervolgens wordt er gebruikt gemaakt van de LEFT$(X$,N) tekstfunktie om alle namen die met een J beginnen (d.w.z. alle lettertekenrijen waar J het eerste letterteken aan de linkerkant is) apart weer te geven.

```
10 DIM N$(10):Y=1
20 CLS
30 FOR L=1 TO 10
40 INPUT "Naam";N$(L)
50 NEXT L
60 CLS
70 FOR L=1 TO 10
80 LOCATE 2,L:PRINT N$(L)
90 NEXT L
100 FOR L=1 TO 10
110 A$=LEFT$(N$(L),1)
120 IF A$<>"J" THEN 150
130 LOCATE 15,Y:PRINT N$(L)
140 Y=Y+1
150 NEXT L
160 LOCATE 0,20
```

Het INPUT bevel in regel 40 wijst namen (van personen) aan de rij-variabelen N$(1) t/m N$(10) toe. Je zou bijvoorbeeld de volgende namen kunnen toewijzen, waarbij je begint bij N$(1):
PETER, PAUL, JAN, MARIE, SIMONE, JOOST, JANET, TOM, DICK, CARIN

Nadat alle namen door de regels 70 t/m 90 zijn weergegeven, zorgen de regels 100 t/m 150 ervoor dat alle namen die beginnen met een J— JAN, JOOST en JANET—aan de rechterkant van het scherm worden weergegeven.

```
PETER       JAN
PAUL        JOOST
JAN         JANET
MARIE
SIMONE
JOOST
JANET
TOM
DICK
CARIN
```

In regel 110 wordt gebruik gemaakt van het LEFT$(X$,N) bevel om alleen die namen die met een J beginnen aan de rechterkant van het scherm weer te geven. Hierbij wordt alleen het eerste letterteken aan de linkerkant van de lettertekenrij die aan de rij-variabele N$ is toegewezen, aan de A$ variabele toegewezen. Vervolgens kontroleert het IF—THEN bevel in regel 20 of dit letterteken "J" is. Wanneer dit het geval is, wordt de inhoud van N$ aan de rechterkant van het scherm weergegeven.

FUNKTIES VOOR HET OMZETTEN VAN NUMERIEKE EN TEKSTGEGEVENS

- Omzettingsfunkties
- Lettertekencodes

DE OMZETTINGSFUNKTIES

Er zijn twee soorten omzettingsfunkties: bij de ene soort worden er getalswaarden als (numerieke) gegevens ingevoerd en wordt de uitkomst gevormd door een rijtje letters of cijfers (lettertekenrij). Bij de andere soort worden er lettertekenrijen ingevoerd en zijn numerieke gegevens de uitkomst. Er bestaan bij MSX2-BASIC negen omzettingsfunkties.

ASC(X$), CHR$(X)
VAL(X$), STR$(X)
LEN(X$), INSTR([N,]X$, Y$)
BIN$(X), OCT$(X), HEX$(X)

WIJZIGEN VAN DE GETALSOORTEN VAL(X$), STR$(X)

In BASIC zijn er twee getalsoorten: numerieke getallen en lettertekenrijgetallen. Als bijvoorbeeld 123 aan een numerieke variabele wordt toegewezen, zoals dat het geval is in

```
A=123
```

dan wordt 123 beschouwd als het getal honderddrieëntwintig. Wanneer 123 echter aan een rijvariabele wordt toegewezen, zoals dat het geval is in

```
A$="123"
```

dan wordt 123 beschouwd als de "lettertekens" een, twee en drie.

VAL(X$) verandert een rijtje cijfers in een getal.
STR$(X) verandert cijfers die samen een getal vormen in een rijtje losse cijfers.

Voer het volgende programma in:

```
10 A$="123":B$="456"
20 X=VAL(A$):Y=VAL(B$)
30 PRINT "A$+B$=";A$+B$
40 PRINT "VAL(A$)+VAL(B$)=";X+Y
```

Allereerst worden de getallen 123 en 456 als een rijtje losse cijfers aan de A$ en de B$ variabelen toegewezen. Vervolgens worden deze in regel 20 omgezet in een numerieke waarde (een getal). De regels 30 en 40 geven het samenvoegen van de rijtjes losse cijfers en het optellen van de cijfers die getallen vormen, weer. In het onderstaande blokje wordt getoond hoe het scherm er bij uitvoering van het programma uit zal zien.

```
RUN
A$+B$=123456
VAL(A$)+VAL(B$)= 579
Ok
```

STR$(X) is de funktie waarvan de werking tegengesteld is aan die van VAL(A$). Voer het volgende programma uit:

```
10 A=123:B=456
20 X$=STR$(A):Y$=STR$(B)
30 PRINT "A+B=";A+B
40 PRINT "STR$(A)+STR$(B)=";X$+Y$
RUN
A+B= 579
STR$(A)+STR$(B)= 123 456
OK
```

Bij het omzetten van cijfers die samen een getal vormen in een rijtje losse cijfers, wordt de ruimte voor het getal (daar waar het + of − teken verschijnt) opgenomen in het rijtje losse cijfers.

LETTERTEKENCODES EN FUNKTIES ASC(X$), CHR$(X)

Net zoals er codes voor de kleuren zijn, zijn er ook codes voor de lettertekens die bij BASIC gebruikt worden. De lettertekencode voor de hoofdletter A is bijvoorbeeld 65 (decimaal getal).
Er zijn bij MSX2-BASIC twee funkties die gebruik maken van de lettertekencodes.

ASC(X$) geeft de lettercode die korrespondeert met een ingevoerd letterteken als uitkomst.
CHR$(X) geeft het letterteken dat korrespondeert met een ingevoerde lettertekencode als uitkomst.

Je kunt deze funkties buiten een programma om als volgt kontroleren:

```
PRINT ASC("A")
 65
Ok
PRINT CHR$(66)
B
Ok
```

Zie achterin het MSX2-BASIC HANDBOEK VOOR HET PROGRAMMEREN voor een lijst van de lettertekencodes.

DE LENGTE VAN EEN LETTERTEKENRIJ BEPALEN LEN(X$)

De LEN(X$) funktie geeft het aantal lettertekens dat zich in lettertekenrij X$ bevindt als een getal (numerieke waarde) als uitkomst.

```
A$="ABCDE"
Ok
PRINT LEN(A$)
 5
Ok
```

Het aantal lettertekens in de lettertekenrij "ABCDE" is 5. LEN(A$) geeft dus 5 als uitkomst.

In het volgende programma wordt gebruik gemaakt van de LEN(X$), MID$(X$,M,N) en de CHR$(X) funktie. Deze funkties zetten ieder letterteken dat met behulp van het INPUT bevel aan de A$ variabele is toegewezen, om in het letterteken dat korrespondeert met de lettertekencode die één hoger is dan de lettertekencode van het ingevoerde letterteken.

```
10 CLS
20 INPUT "Wat letters";A$
30 N=LEN(A$)
40 FOR L=1 TO N
50 B$=MID$(A$,L,1)
60 X=ASC(B$)
70 C$=CHR$(X+1)
80 PRINT C$;
90 NEXT L
100 END
```

Probeer het maar eens door lettertekens in te voeren en bekijk het resultaat.

```
RUN
Wat letters? RNMX
SONY
Ok
RUN
Wat letters? LRW
MSX
Ok
```

FUNKTIES VOOR DE INVOER VAN GEGEVENS

- Wat zijn gegevensinvoer-funkties?
- Invoeren van gegevens via het toetsenbord
- Invoeren van de stand van de cursortoetsen

HET GEBRUIK VAN DE GEGEVENSINVOER-FUNKTIES

De funkties die we tot nu toe hebben besproken, zijn allemaal funkties waarbij er een bepaalde waarde ingevoerd wordt. Deze waarde wordt door de funktie bewerkt, hetgeen resulteert in een bepaalde uitkomst. De funkties die in die gedeelte behandeld worden verschillen enigzins van de eerder behandelde funkties.
Bij deze funkties worden de ingevoerde waarden niet direkt bewerkt. De bij deze funkties ingevoerde waarden lijken op tekens die een speciale betekenis hebben. Hierdoor worden gegevens met betrekking tot de status van de voor invoer aangesloten apparaten als uitkomst gegeven. (Bij sommige van deze gegevensinvoer-funkties is het niet nodig om bepaalde waarden in te voeren.)

Het gebruik van deze funkties maakt het mogelijk om een programma te schrijven dat een handeling uitvoert, gebaseerd op de status van een voor invoer aangesloten apparaat. Bijvoorbeeld een programma om een beeldpatroon in de richting van een ingedrukte cursortoets te bewegen.

De gegevensinvoer-funkties

MSX2-BASIC beschikt over 22 gegevensinvoer-funkties

- Invoer via het scherm
  CSRLIN, POS(X), POINT(X,Y)
- Invoer via de afdrukker
  LPOS(X)
- Invoer via het geheugen
  FRE(X), FRE(" "), PEEK(N), VARPTR(variabele), VPEEK(N)
- Invoer via het toetsenbord
  INKEY$, INPUT$(X)
- Invoer via een I/O poort
  INP(N)
- Invoer via het spelpookje, de spatiebalk, muis, volgbal, peddel, aanraakpaneel of lichtpen.
  STICK(N), STRIG(N), PDL(N), PAD(N)
- Invoer via een (gegevens) bestand
  EOF(bestandsnummer), INPUT$(N,[#]bestandsnummer)
- Invoer via een diskette
  DSKF(nummer diskette-eenheid), LOC(bestandsnummer),
  VARPTR(#bestandsnummer)
- Invoer via een subroutine in machinetaal
  USR[X](I)

GEGEVENS INVOEREN VIA HET TOETSENBORD

INKEY$

Bij het indrukken van een lettertekentoets op het toetsenbord, geeft de INKEY$ funktie als uitkomst de betreffende letter (of het cijfer) als rij-waarde.
Het volgende eenvoudige programma laat het gebruik van de INKEY$ funktie zien.

```
10 K$=INKEY$
20 PRINT K$;
30 GOTO 10
```

Bij uitvoering van dit programma

```
K$=INKEY$
```

zal regel 10 telkens opnieuw uitgevoerd worden. Wanneer er bij uitvoering van regel 10 geen toets op het toetsenbord wordt ingedrukt, geeft de INKEY$ funktie een lettertekenrij die geen gegevens bevat als uitkomst. Een dergelijke lettertekenrij wordt een **lege rij** genoemd. In het bovenstaande programma wordt de uitkomst van de INKEY$ funktie aan K$ toegewezen, ook als het een lege rij betreft. Het PRINT bevel in regel 20 geeft de rij op het scherm weer. Om dat het in dit geval om een lege rij gaat, zal het er op het scherm uitzien alsof er niets gebeurd is.
Wanneer er tijdens het uitvoeren van regel 10 een toets—zoals [A]—wordt ingedrukt, zal de INKEY$ funktie A als uitkomst geven. (Bij het indrukken van [a], is de uitkomst a.) Dit letterteken is toegewezen aan K$. K$ wordt door regel 20 op het scherm weergegeven en er verschijnt dus een A op het scherm.

```
RUN
A
```

Bij indrukken van de [A] toets.

Wanneer er nog andere toetsen worden ingedrukt, dan worden deze lettertekens ook weergegeven. Het scherm ziet er hetzelfde uit als bij de wachtstand, alleen is in dit geval de cursor niet weergegeven. Dit duidt er op dat er een programma wordt uitgevoerd. Wanneer de [↵] toets wordt ingedrukt, dan wordt het volgende letterteken aan het begin van dezelfde regel weergegeven in plaats van op de volgende regel.
Met behulp van [CTRL]+[STOP] kun je het programma beëindigen.

Een ander programma dat gebruik maakt van de INKEY$ funktie:

```
10 CLS
20 LOCATE 5,4
30 PRINT "┌───────────────┐"
40 LOCATE 5,14
50 PRINT "└───────────────┘"
60 FOR Y=5 TO 13 STEP 2
70 FOR X=6 TO 20
80 K$=INKEY$
90 IF K$="" THEN 80
100 LOCATE X,Y:PRINT K$
110 NEXT X
120 NEXT Y
130 LOCATE 0,22:END
```

Dit programma laat een van de belangrijkste gebruiksmogelijkheden van de INKEY$ funktie zien. Bekijk eens de kombinatie van het INKEY$ bevel in regel 80 met het IF—THEN bevel in regel 90.

```
80 K$=INKEY$
90 IF K$="" THEN 80
```

Wanneer er bij de uitvoering van regel 80 geen toets wordt ingedrukt, dan wordt er een lege rij aan K$ toegewezen. Wanneer K$ uit een lege rij bestaat, dan laat regel 90 het programma terugkeren naar regel 80. Daarom blijft het programma een lus herhalen, totdat er een toets ingedrukt wordt.
Wanneer er zich geen letterteken of spatie tussen de aanhalingstekens (“ ”) bevindt in

```
K$=""
```

dan wijst dat op een lege rij.
Bij het indrukken van een toets gaat het programma verder naar regel 100 en het letterteken dat bij de ingedrukte toets hoort, wordt weergegeven.

De INKEY$ funktie wordt vaak op deze manier gebruikt om een programma bij indrukken van een toets verder te laten gaan naar de volgende stap.

Je kunt de INKEY$ funktie ook gebruiken om bij het indrukken van een speciale toets het programma verder te laten gaan naar de volgende stap. In het volgende programma is de spatiebalk die speciale toets.

```
10 CLS
20 INPUT "Wat letters";A$
30 K$=INKEY$
40 IF K$="" OR K$<>" " THEN 30
50 PRINT A$
```

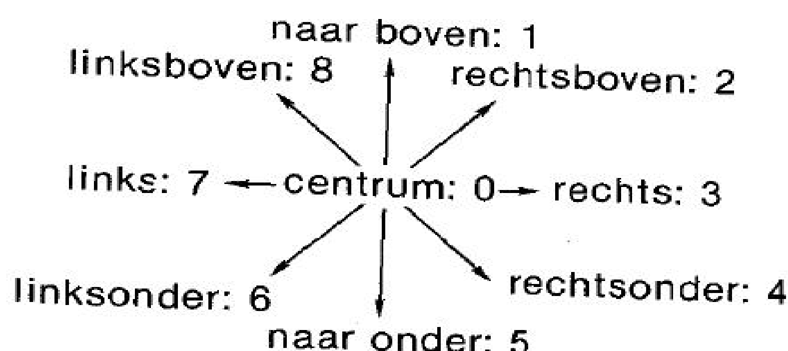
Het IF—THEN bevel in regel 40 laat het programma terugkeren naar regel 30, wanneer:
1) K$ uit een lege rij bestaat;
2) Wanneer K$ niet de spatiebalk is. Het programma gaat alleen verder naar regel 50 wanneer de spatiebalk ingedrukt wordt.

INVOEREN VAN DE STAND VAN DE CURSORTOETSEN STICK(N)

De STICK(N) funktie geeft als uitkomst een numerieke waarde die de richting van de cursortoets, het spelpookje, de volgbal of het aanraakpaneel aangeeft.
De waarde die aan N wordt toegekend bepaalt of de stand van de cursortoetsen of de stand van een van de andere voor invoer aangesloten apparaten als uitkomst wordt gegeven.

N	Apparaat
0	cursortoets
1	apparaat dat aangesloten is op CONTROLLER A
2	apparaat dat aangesloten is op CONTROLLER B

STICK(N) geeft als uitkomst een cijfer tussen 0 en 8. Deze geeft de richting van de cursortoets of een van de andere apparaten aan.



Als er bijvoorbeeld geen cursortoets ingedrukt wordt, dan zal STICK(0) 0 als uitkomst geven.
Bij het indrukken van de (naar boven) toets wordt er 1 als uitkomst gegeven, indrukken van de (naar beneden) toets geeft 3 en tegelijkertijd indrukken van de en de toets geeft 2 als uitkomst.

Een programma waarbij de STICK(N) funktie gebruikt wordt om met de cursortoetsen de weergave op het scherm regelen, is het volgende:

```
10 CLS
20 X=14:Y=10
30 LOCATE X,Y:PRINT "o"
40 C=STICK(0)
50 IF C=0 THEN 40
60 IF C=1 THEN VX=0:VY=-1:GOSUB 110
70 IF C=3 THEN VX=1:VY=0:GOSUB 110
80 IF C=5 THEN VX=0:VY=1:GOSUB 110
90 IF C=7 THEN VX=-1:VY=0:GOSUB 110
100 GOTO 40
110 X=X+VX:Y=Y+VY
120 IF X>29 THEN X=29
130 IF X<0 THEN X=0
140 IF Y>21 THEN Y=21
150 IF Y<0 THEN Y=0
160 LOCATE X,Y:PRINT "o"
170 RETURN
```

Bij dit programma wordt de kleine letter "o" weergegeven in de richting die korrespondeert met de beweging van de cursortoets. De "o" wordt eerst door de regels 20,30 op het punt (14,10) weergegeven. De STICK(N) funktie in regel 40 geeft de stand van de cursortoets als uitkomst. De regels 50 t/m 90 bepalen de waarde van de VX en VY variabele. Dit wordt gedaan aan de hand van de uitkomst van de STICK(0) funktie. Op deze manier wordt de volgende plaats bepaald waar de "o" zal worden weergegeven.
Wanneer bijvoorbeeld de → toets ingedrukt wordt, geeft de STICK(0) funktie 3 als uitkomst en in regel 70 wordt aan VX de waarde 1 en aan VY de waarde 0 toegekend. Vervolgens verspringt het programma naar de subroutine, die aan het begin van regel 110 staat.

In de subroutine worden aan X en Y de coördinaten toegewezen van het volgende punt waar "o" wordt weergegeven zal worden. Hierna wordt "o" weergegeven. De regels 120 t/m 150 in de subroutine beperken de afmetingen van het weergavegebied.

Hoofdstuk 8
Onderbrekingen

MAKEN VAN ONDERBREKINGEN

- Wat is een onderbreking?
- MSX2-BASIC onderbrekingen
- Onderbrekingen maken

WAT IS EEN ONDERBREKING?

Een onderbreking wordt veroorzaakt door een ingreep, naar keuze, van buitenaf en dient om een programma tijdens de verwerking tijdelijk te stoppen, om tussentijds andere bewerkingen uit te voeren. De reeks van handelingen die worden uitgevoerd na het onderbreken van een programma wordt een onderbrekingsverwerkingsprogramma of onderbrekingsroutine genoemd.
Een ingreep die soortgelijk is aan de onderbreking is de subroutine. Een subroutine wordt echter uitsluitend gestart door een GOSUB bevel. Met andere woorden, een subroutine is van tevoren gepland en vindt beheel binnen het programma plaats.
Voor het veroorzaken van een onderbreking is echter altijd een externe ingreep nodig (zoals bijvoorbeeld het indrukken van de F1 toets).
Na het beëindigen van een onderbrekingsroutine wordt de verwerking van het hoofdprogramma gewoonlijk op dezelfde wijze hervat als na het uitvoeren van een subroutine.

MSX2-BASIC ONDERBREKINGEN

MSX-BASIC beschikt over een vijftal manieren om een onderbreking te maken. Een onderbreking kan worden ingeschakeld door:
- Indrukken van een funktietoets (F1—F10).
- Indrukken van de spatiebalk of de treftoets of vuurknop van een spelpookje, muis, volgbal of aanraakpaneel.
- Indrukken van CTRL + STOP, tegelijk.
- Overlappen van beeldpatronen (sprites).
- Het verstrijken van een bepaalde periode (afgemeten aan de ingebouwde klok).

ONDERBREKINGEN INSCHAKELEN

De vijf instrukties in de rechter kolom van onderstaande tabel dienen om binnen de hoofdroutine van een MSX2-BASIC programma een onderbreking van te voren aan te kondigen, om bij de verwerking van het programma rechtstreeks over te kunnen gaan tot het uitvoeren van een onderbrekingsroutine.

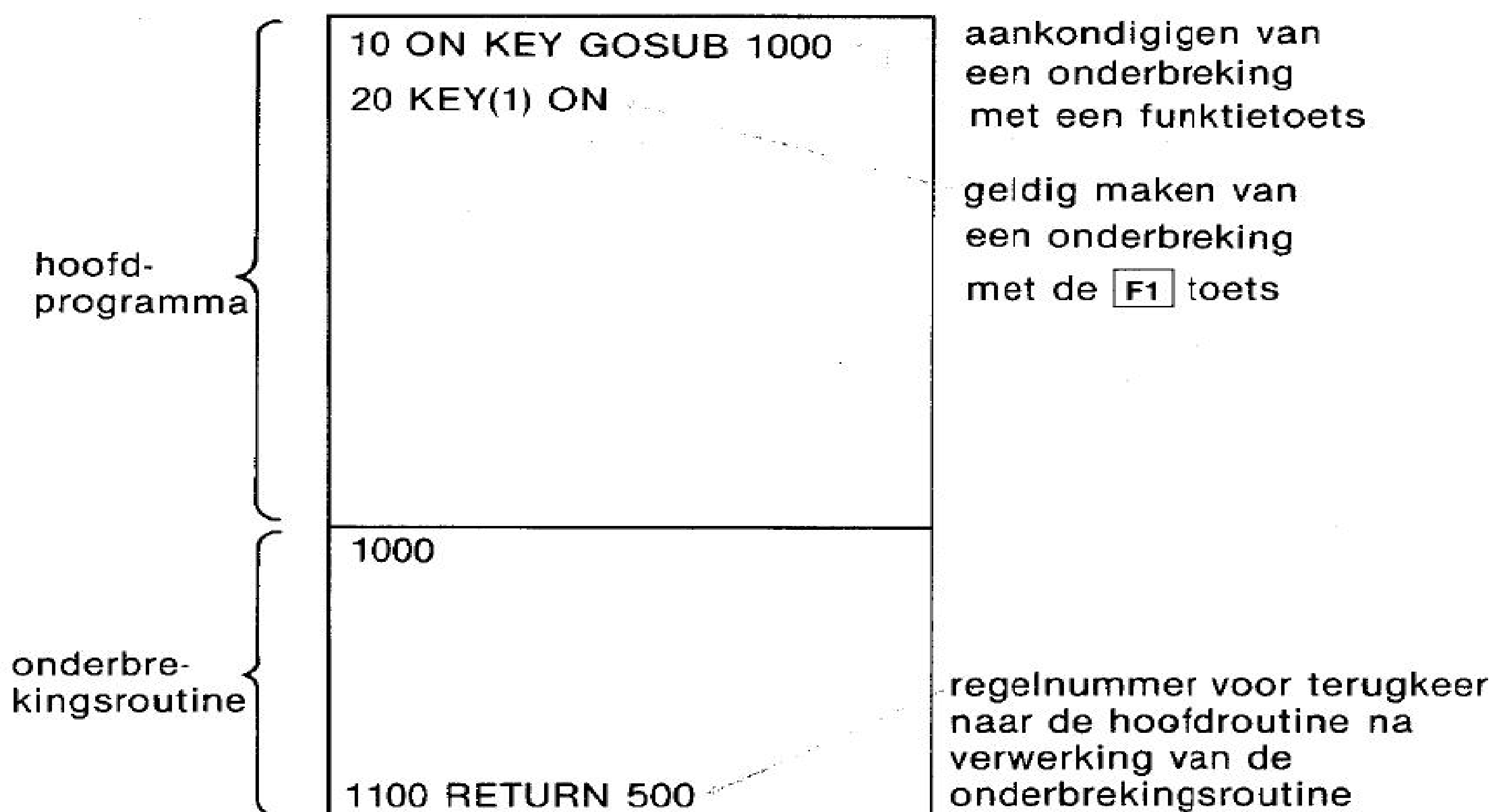
Een onderbreking kan worden ingeschakeld door:	Instruktie om onderbreking aan te kondigen
Indrukken van een funktietoets	ON KEY GOSUB regelnummer [, regelnummer]...
Indrukken van spatiebalk of trekkerknop van een spelpookje, muis, volgbal of aanraakpaneel	ON STRIG GOSUB regelnummer [, regelnummer]...
Indrukken van CTRL + STOP	ON STOP GOSUB regelnummer
Overlappen van beeldpatronen	ON SPRITE GOSUB regelnummer
Het verstrijken van een periode op de ingebouwde klok	ON INTERVAL = periode GOSUB regelnummer

De instruktie voor aankondigen van een onderbreking geeft aan waardoor de onderbreking veroorzaakt wordt en geeft bovendien het nummer van de beginregel van de onderbrekingsroutine.

Na deze instruktie volgt onmiddellijk nog een tweede, die dient voor het geldig maken van de onderbreking. Er zijn weer vijf verschillende instrukties waarmee je een onderbreking geldig kunt maken.

Onderbreking in te schakelen door:	Instruktie om onderbreking geldig te maken
Indrukken van een funktietoets	KEY(N) ON (N kan 1 t/m 10 zijn, waarbij 1 staat voor de F1 toets)
Indrukken van spatiebalk of trekkerknop van een spelpookje, muis, volgbal of aanraakpaneel	STRIG(N) ON (N is 0 t/m 4, waarbij 0 staat voor de spatiebalk)
Indrukken van CTRL + STOP	STOP ON
Overlappen van beeldpatronen	SPRITE ON
Het verstrijken van een periode op de ingebouwde klok	INTERVAL ON

Bij uitvoeren van het volgende programma zal door het indrukken van funktietoets F1 de verwerking doorgaan met de onderbrekingsroutine die begint bij regel 1000.



Bij uitvoeren van dit programma wordt normaal de hoofdroutine afgewerkt, maar bij indrukken van de F1 toets tijdens verwerking verspringt het programma naar regel 1000, waarna de onderbrekingsroutine wordt uitgevoerd. Na volledige verwerking hiervan stuurt het RETURN 500 bevel aan het eind van de onderbrekingsroutine de computer terug naar regel 500 van het hoofdprogramma.

PROGRAMMA'S MET ONDERBREKINGEN

- Onderbrekingen met funktietoetsen
- Ongeldig maken van een onderbreking
- Vasthouden van een onderbreking
- Onderbrekingen door overlappen van beeldpatronen

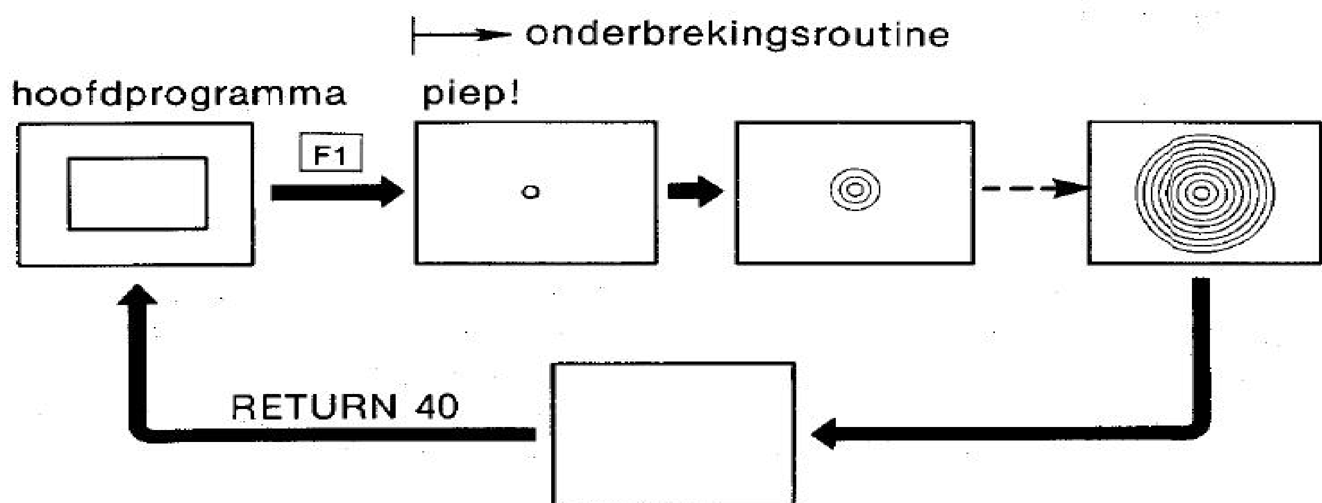
EEN PROGRAMMA ONDERBREKEN MET EEN FUNKTIETOETS

Het volgende programma laat zien hoe de [F1] funktietoets gebruikt kan worden om een onderbreking in te schakelen.

```
10 ON KEY GOSUB 100
20 KEY(1) ON
30 SCREEN 2
40 LINE (50,50)-(200,150),,B
50 GOTO 40
100 'subroutine
110 BEEP:CLS
120 FOR L=10 TO 90 STEP 10
130 CIRCLE (120,100),L
140 NEXT L
150 CLS
160 RETURN 40
```

(regels 10–50: hoofd-programma; regels 100–160: onderbrekings-routine)

Dit programma is door het invoegen van regel 10 en 20 zo opgezet dat bij het indrukken van de [F1] funktietoets wordt doorgegaan met een subroutine die loopt vanaf regel 110.
Bij het verwerken van dit programma wordt door regel 40 en 50 van het hoofdprogramma onafgebroken een rechthoek op het scherm afgebeeld. Wanneer echter de [F1] toets wordt ingedrukt vindt een onderbreking plaats, waarna volgens het bevel van regel 10 de verwerking doorgaat met regel 100. Het merkbare resultaat is dat de rechthoek verdwijnt met een pieptoon (BEEP:CLS) en dat er 9 cirkels op het scherm worden getekend. Na het trekken van de laatste cirkel wordt het scherm gewist en verschijnt weer de rechthoek van regel 40 en 50.


onderbrekingsroutine
hoofdprogramma
piep!
F1
RETURN 40

ONGELDIG MAKEN VAN EEN ONDERBREKING KEY(N) OFF

Laten we aan het bovenstaande programma eens de volgende regel toevoegen.

```
105 KEY(1) OFF
```

Verwerk het programma. Bij het eenmaal indrukken van de F1 toets vindt een onderbreking plaats. Daarna volgt op het nogmaals indrukken van de F1 toets geen enkele onderbreking meer.
De oorzaak hiervan is gelegen in het feit dat na het eenmaal uitvoeren van de onderbrekingsroutine het bevel van regel 105

```
KEY(1) OFF
```

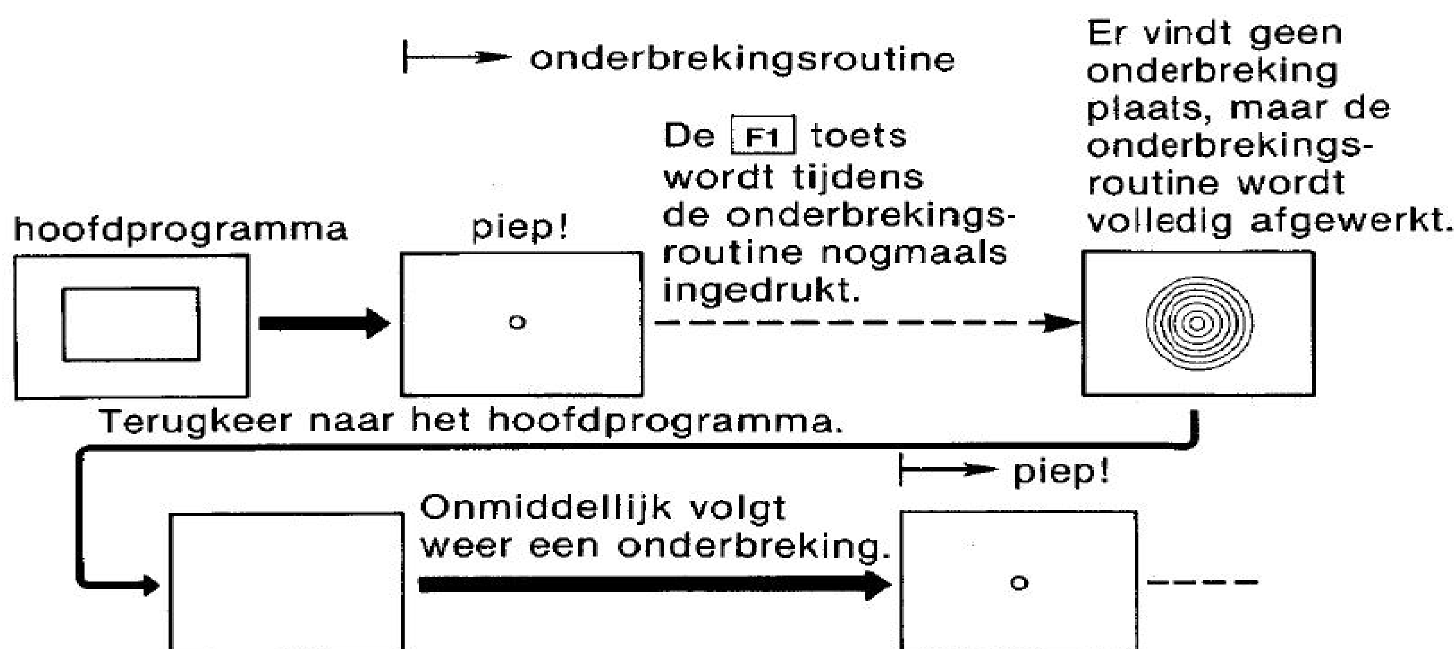
volgde, waarmee de onderbreking met de F1 toets ongeldig werd gemaakt.
De KEY(N) OFF instruktie dient voor het ongeldig maken van een onderbreking. N staat voor het nummer van de in te drukken funktietoets.

De volgende tabel geeft de vijf MSX2-BASIC instrukties waarmee je een onderbreking ongeldig kunt maken.

Onderbreking in te schakelen door:	Instruktie om onderbreking ongeldig te maken
Indrukken van een funktietoets	KEY(N) OFF
Indrukken van spatiebalk of trekkerknop van een spelpookje, muis, volgbal of aanraakpaneel	STRIG(N) OFF
Indrukken van CTRL + STOP	STOP OFF
Overlappen van beeldpatronen	SPRITE OFF
Het verstrijken van een periode op de ingebouwde klok	INTERVAL OFF

VASTHOUDEN VAN EEN ONDERBREKING

Wanneer bij een onderbreking de verwerking van een programma wordt overgebracht naar een onderbrekingsroutine, treedt de zogenaamde vasthoudstand van een onderbreking op. Wanneer in deze stand opnieuw getracht wordt de verwerking te onderbreken heeft dit geen direkt effekt, maar voor terugkeer naar het hoofdprogramma met een RETURN bevel wordt automatisch een ·ON bevel ingelast. Het hoofdprogramma wordt in dat geval nog niet uitgevoerd, maar er wordt snel opnieuw overgegaan tot uitvoeren van de onderbrekingsroutine.
Anders gezegd, tijdens de onderbrekings-vasthoudstand wordt nooit tussentijds teruggegaan naar de beginregel van de onderbrekingsroutine, maar wordt een gewenste onderbreking vastgehouden tot de onderbrekingsroutine geheel beëindigd is, waarna deze in zijn geheel herhaald wordt.
Voor het programma op blz. 218 betekent dit dat bij éénmaal indrukken van de [F1] toets door de onderbrekingsroutine 9 cirkels worden getekend. Als echter voor het kompleteren van de laatste cirkel opnieuw de [F1] toets wordt ingedrukt gebeurt er ogenschijnlijk niets. Na de laatste cirkel wordt teruggegaan naar het hoofdprogramma. Dan vindt echter wel opnieuw een onderbreking plaats, voor de tweede maal de [F1] toets is ingedrukt, en er worden in plaats van de rechthoek opnieuw 9 cirkels getekend.

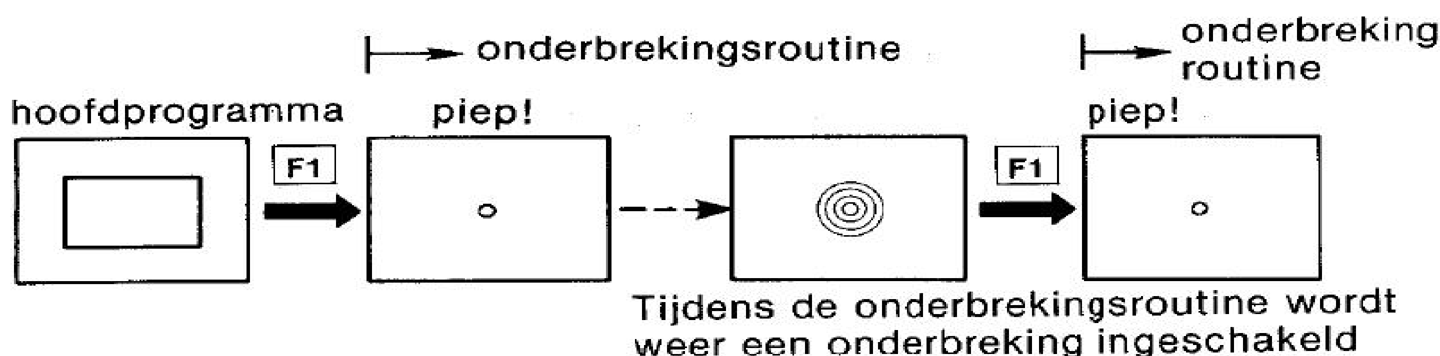


GELDIG MAKEN VAN EEN ONDERBREKING TIJDENS EEN ONDERBREKINGSROUTINE KEY(N) ON

Om een onderbreking ook in de vasthoudstand, dus tijdens het uitvoeren van een onderbrekingsroutine, geldig te maken geef je een voor geldig maken bestemd bevel als KEY(1) ON. Het resultaat is dan dat als een onderbrekingsroutine op zijn beurt onderbroken wordt, deze onmiddellijk vanaf het begin herhaald zal worden.

```
10 ON KEY GOSUB 100
20 KEY(1) ON
30 SCREEN 2
40 LINE (50,50)-(200,150),,B
50 GOTO 40
100 'subroutine
105 KEY(1) ON
110 BEEP:CLS
120 FOR L=10 TO 90 STEP 10
130 CIRCLE (120,100),L
140 NEXT L
150 CLS
160 RETURN 40
```

Dit programma is vrijwel gelijk aan het vorige, met als enige wijziging het KEY(1) ON bevel dat in regel 105 is ingevoegd. Als gevolg van deze toevoeging brengt ook tijdens het tekenen van de cirkels het indrukken van de F1 toets onmiddellijk een onderbreking teweeg, waarna de onderbrekingsroutine die begint met regel 100 in zijn geheel opnieuw wordt gestart.

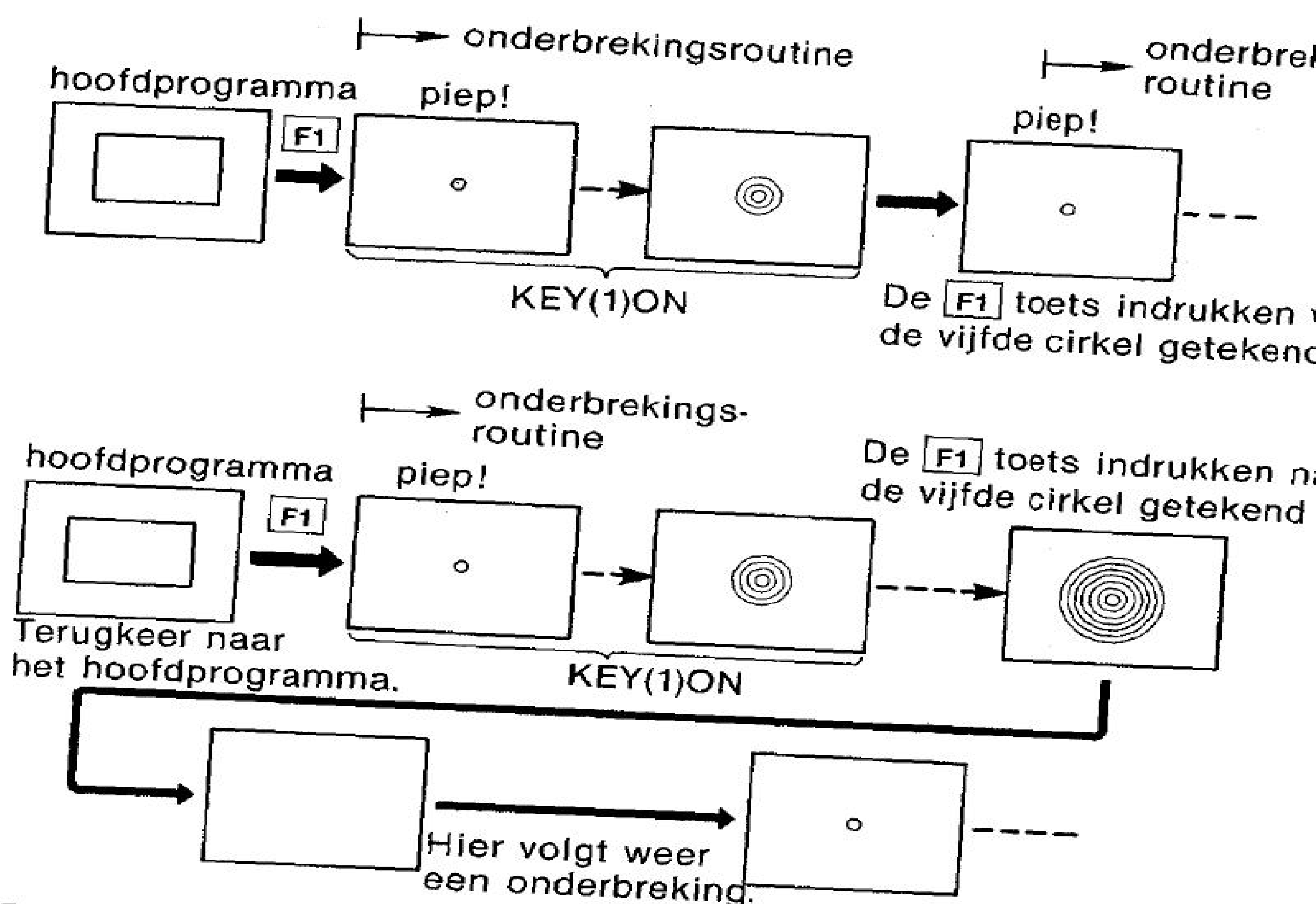


VASTHOUDEN VAN EEN ONDERBREKING IN EEN PROGRAMMA KEY(N) STOP

Na het geldig maken van een onderbreking tijdens de onderbrekings-routine met een KEY(N) ON bevel is het mogelijk om tijdelijk een vast-houdstand in te stellen door het invoegen van een KEY(N) STOP bevel.

```
10 ON KEY GOSUB 100
20 KEY(1) ON
30 SCREEN 2
40 LINE (50,50)-(200,150),,B
50 GOTO 40
100 'subroutine
105 KEY(1) ON
110 BEEP:CLS
120 FOR L=10 TO 90 STEP 10
130 CIRCLE (120,100),L
135 IF L=50 THEN KEY(1) STOP
140 NEXT L
150 CLS
160 RETURN 40
```

Dit programma is weer bijna gelijk aan het vorige, maar deze keer met de extra bijzonderheid dat wanneer L de waarde 50 bereikt, in regel 135 de KEY(1) STOP instruktie in werking treedt. Zodoende vindt bij het indrukken van de F1 toets vóór het tekenen van de vijfde cirkel onmiddellijk weer een onderbreking plaats, terwijl bij het indrukken na het tekenen van de vijfde cirkel de vasthoudstand wordt ingesteld, zodat de onderbreking pas later volgt.



De KEY(N) STOP instruktie zorgt dat bij indrukken van een funktietoets de vasthoudstand wordt ingesteld. N staat voor het nummer van de funktietoets.
De volgende tabel geeft de vijf MSX2-BASIC instrukties waarmee je een onderbreking tijdelijk kunt vasthouden.

Onderbreking in te schakelen door:	Instruktie om onderbreking vast te houden
Indrukken van een funktietoets	KEY(N) STOP
Indrukken van spatiebalk of trekkerknop van een spelpookje, muis, volgbal of aanraakpaneel	STRIG(N) STOP
Indrukken van CTRL + STOP	STOP STOP
Overlappen van beeldpatronen	SPRITE STOP
Het verstrijken van een periode op de ingebouwde klok	INTERVAL STOP

ONDERBREKING DOOR OVERLAPPEN VAN BEELDPATRONEN

Zodra twee of meer beeldpatronen elkaar met minimaal 1 stip overlappen kan hierop een onderbreking volgen door de ON SPRITE GOSUB en SPRITE ON bevelen.
In het volgende programma komen UFO's zowel van links als van rechts gevlogen, en wanneer ze elkaar overlappen klinkt een pieptoon.

```
10 SCREEN 2
20 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR$
(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 ON SPRITE GOSUB 100
40 SPRITE ON
50 FOR X=0 TO 255
60 PUT SPRITE 0,(X,100),15,0
70 PUT SPRITE 1,(255-X,100),10,0
80 NEXT X
90 END
100 SPRITE OFF
110 BEEP
120 SPRITE ON
130 RETURN
```

Programmeervoorbeeld

In het volgende programma gaan we boogschieten, waarbij een onderbreking volgt wanneer de F1 funktietoets wordt ingedrukt, en ook wanneer twee geeldpatronen elkaar overlappen.
Bij indrukken van de F1 toets wordt een pijl afgeschoten. Het aantal punten dat wordt toegekend is afhankelijk van waar de pijl het doelwit raakt. Er kunnen vijf pijlen geschoten worden. Als het totale puntental de 1000 overschrijdt, krijg je een vrij spel.

```
10 SCREEN 5,1
20 OPEN "GRP:" FOR OUTPUT AS #1
30 ON KEY GOSUB 250 : ON SPRITE GOSUB 27
0
40 ' *** beeldpatronen ***
50 RESTORE 430:SN=0:GOSUB 310
60 RESTORE 520:SN=1:GOSUB 310
70 ' *** hoofdprogramma ***
80 X=230:S=0:W=50
90 FOR L=1 TO 5
100  SPRITE  ON:KEY (1) ON
110  U=W:M=0:B=0:V=106:GOSUB 210
120  FOR Y=0 TO 211
130   PUT SPRITE 0,(X,Y),,0
140   U=U+B:IF M=1 THEN V=V+1
150   PUT SPRITE 1,(U,V),,1
160   IF U>250 THEN U=W:B=0
170  NEXT Y
180 NEXT L
190 IF S>=1000 THEN 80 ELSE END
200 ' *** scorebord ***
210 PRESET (30,10)
220 PRINT #1,USING "## : ####";L,S
230 RETURN
240 ' *** schieten
250 KEY (1) OFF:B=3:RETURN
260 ' *** raak ***
270 SPRITE OFF:B=0:M=1
280 S=S+(8-ABS(Y-V+1))^2*10
290 GOSUB 210 : RETURN
300 ' *** beeldpatronen subroutine ***
310 SP$="":SC$=""
320 FOR SL=0 TO 7
330 READ SQ$,SC:SP=0
340  FOR SJ=1 TO 8
350   SP=SP*2-(MID$(SQ$,SJ,1)="O")
360  NEXT SJ
370  SP$=SP$+CHR$(SP):SC$=SC$+CHR$(SC)
380 NEXT SL
390 SPRITE$(SN)=SP$
400 COLOR SPRITE$(SN)=SC$
410 RETURN
420 ' *** gegevens beeldpatroon ***
430 DATA ......OO,1
440 DATA ......OO,8
450 DATA ......OO,15
460 DATA ......OO,8
470 DATA ......OO,8
480 DATA ......OO,15
490 DATA ......OO,8
500 DATA ......OO,1
```

samenstellen beeldpatroon

subroutine voor geven van het puntental

gegevens voor beeldpatroon (doelwit)

```
510 ' *** gegevens beeldpatroon ***
520 DATA .........,0
530 DATA .........,0
540 DATA .........,0
550 DATA OOOOOOOO,15
560 DATA .........,0
570 DATA .........,0
580 DATA .........,0
590 DATA .........,0
```

Het OPEN bevel in regel 20 dient om lettertekens op het grafische scherm te kunnen afbeelden. Dit wordt nader uitgelegd in hoofdstuk 9.

Hoofdstuk 9
Werken met bestanden

BESTANDEN EN APPARATEN

- Wat zijn bestanden?
- Bestandsverwerkende apparaten
- Verwerken van programmabestanden
- Beheer van bestanden

BESTANDEN EN BESTANDSNAMEN

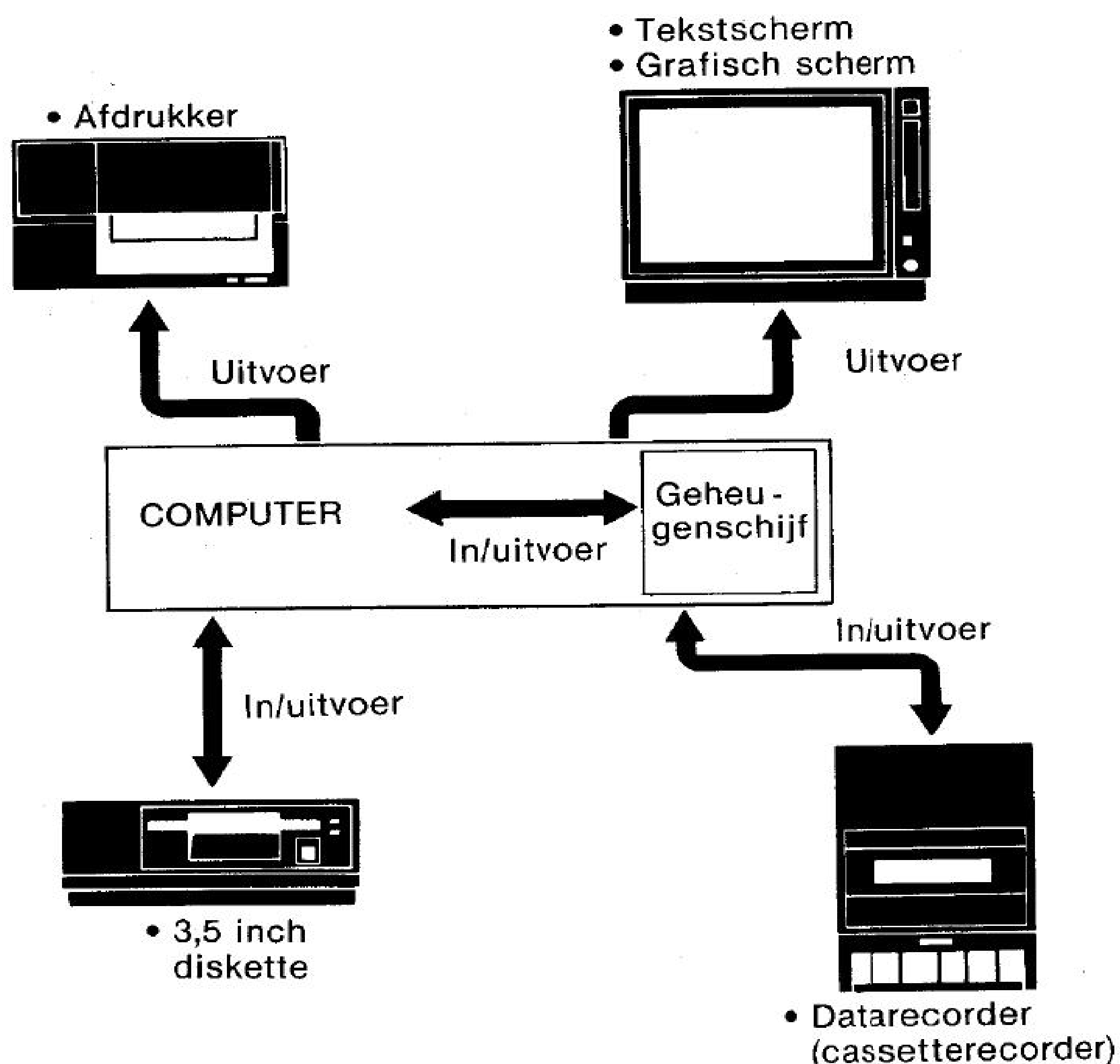
Het komt veelvuldig voor dat een blok gegevens in een programma wordt uitgewisseld tussen de computer en hierop aangesloten apparaten. Dit is wel wat te vergelijken met het vinden van een boek op een boekenplank. Stel dat je een dagboek bijhoudt. Je hebt in je kamer een aantal boekenplanken en op een ervan staat een notitieboek getiteld "dagboek". Voor het lezen of schrijven in je dagboek ga je naar de juiste boekenplank en neemt het betreffende notitieboek van de plank.

Een computer gaat op soortgelijke wijze te werk voor het lezen of schrijven van een programma of een blok gegevens, te vergelijken met de inhoud van het dagboek. Het notitieboek waarin de gegevens komen te staan heet in computerterminologie een bestand. Een bestand verschilt in zoverre van een notitieboek, dat je een bestand in werkelijkheid niet kunt aanraken en oppakken. Het is in feite een bepaald gedeelte op een cassettebandje of op een diskette.

Bestandsnamen
De titel "dagboek" waar we naar zoeken op de plank is bij de computer de naam die we aan een bestand gegeven hebben, de **bestandsnaam**.

BESTANDSVERWERKENDE APPARATEN EN APPARAATNAMEN

De boekenplanken waar we uit kiezen worden bij de computer gevormd door de aangesloten apparatuur, de bestandsverwerkende apparaten. Er bestaan twee soorten bestandsverwerkende apparaten: in-/uitvoer apparaten (cassetterecorder, diskette-eenheid, e.d.) en apparaten die alleen voor invoer kunnen dienen (afdrukker, videomonitor, e.d.). In MSX2-BASIC zijn bevelen beschikbaar voor het uitwisselen van gegevens tussen de computer en verschillende aangesloten apparaten. Het onderstaande overzicht laat zien welke bestandsverwerkende apparaten er bij MSX2-BASIC gebruikt kunnen worden.



De diskettes gebruik je in de ingebouwde diskette-eenheid of in een externe diskette-eenheid die op de computer aangesloten is.

Apparaatnamen

Bij het uitwisselen van bestanden tussen de computer en de randapparatuur moet in het bevel altijd de naam van het gebruikte apparaat genoemd zijn. Hiervoor beschikt MSX-BASIC over apparaatnamen, die in het onderstaande overzicht te zien zijn.

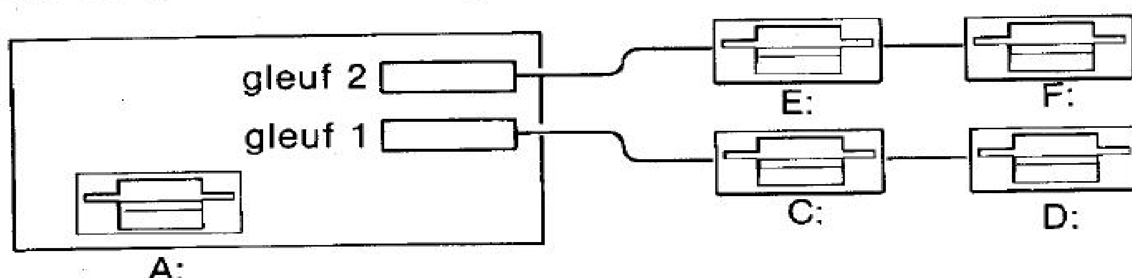
Bestandsverwerkend apparaat	Apparaatnaam
Gegevensrecorder (cassetterecorder)	CAS:
Tekstscherm	CRT:
Grafisch scherm	GRP:
Afdrukeenheid	LPT:
Diskette A diskette-eenheid B diskette-eenheid ⋮ H diskette-eenheid	 A: B: Namen diskette-eenheden ⋮ H:
Geheugenschijf	MEM:

De diskette-eenheid die in de computer ingebouwd is, wordt de A diskette-eenheid genoemd. Wanneer de computer over twee diskette-eenheden beschikt, dan worden deze diskette-eenheid A en B genoemd. Als er een externe diskette-eenheid op de cartridge-gleuf van jouw computer is aangesloten, omdat deze niet ingebouwd is, dan wordt deze diskette-eenheid A genoemd. Mochten er nog andere diekette-eenheden aangesloten zijn, dan zijn dat diskette-eenheid B, C, enz. De apparaatnaam van een dergelijke diskette-eenheid (zoals A:, B:), wordt ook wel diskette-eenheidsnaam genoemd.

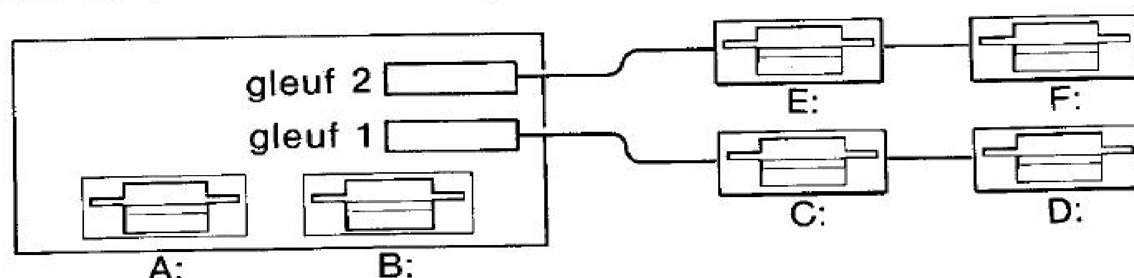
Namen diskette-eenheden

De volgende afbeelding laat de namen van diskette-eenheden zien wanneer je met een kombinatie van de ingebouwde en externe diskette-eenheden (aangesloten op de cartridge-gleuven van de computer) werkt.

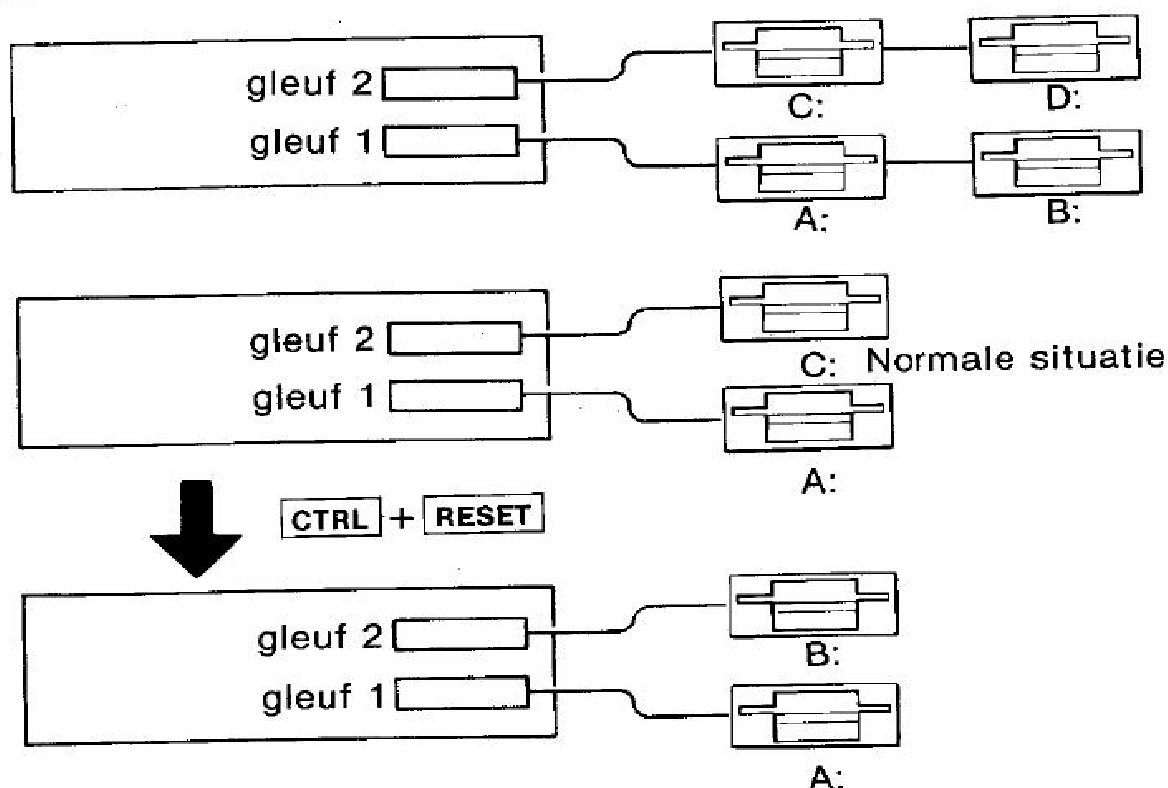
Een computer met een ingebouwde diskette-eenheid



Een computer met twee ingebouwde diskette-eenheden



Een computer die niet voorzien is van ingebouwde diskette-eenheden



REGELS VOOR DE BESTANDS- EN SOORTNAAM

Elk bestand moet een naam hebben bestaande uit een rij van ten hoogste 6 lettertekens (voor een bestand op cassetteband) of 8 lettertekens (voor andere bestanden), beginnend met een letter van het alfabet.
Als je een diskette gebruikt, kun je de bestandsnaam niet weglaten. Dit kan wel bij andere bestandsverwerkende apparaten, maar het verdient aanbeveling altijd bestandsnamen te geven. Op die manier kun je verschillende bestanden van elkaar onderscheiden.
Je kunt een soortnaam aan de bestandsnaam toevoegen door achter de bestandsnaam een punt (.) te zetten en deze te laten volgen door ten hoogste drie lettertekens. De soortnaam wordt gebruikt om de verschillende soorten—zoals BASIC en ASCII—bestanden van elkaar te kunnen onderscheiden. Je kunt de ruimte voor de soortnaam ook gebruiken om een verschil te maken tussen bestanden die met een bestandsnaam van 8 lettertekens niet voldoende van elkaar te onderscheiden zijn.
De volgende voorbeelden tonen twee manieren om de soortnaam te gebruiken.

Voorbeeld 1

TEST.BAS	In dit geval wordt er .BAS aan de bestandsnaam toegevoegd om aan te geven dat het om een BASIC bestand gaat.
TEST.ASC	Hier wordt er .ASC toegevoegd zodat duidelijk is dat het om een bestand gaat dat in ASCII code opgeslagen is.
TEST.DAT	Door .DAT aan de bestandsnaam toe te voegen, wordt aangegeven dat het om een bestand met gegevens (data) gaat.

Voorbeeld 2

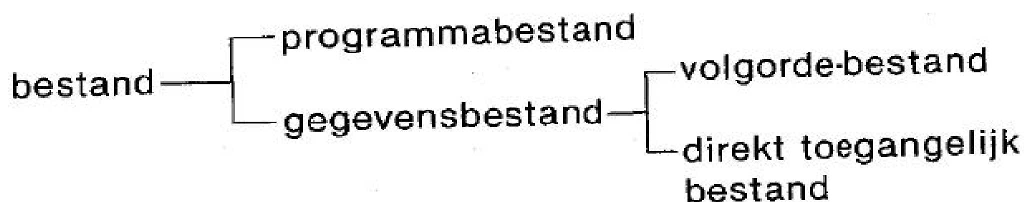
TESTPROG1 TESTPROG2	Omdat de naam TESTPROG1 uit 9 lettertekens bestaat, zal in plaats hiervan TESTPROG als bestandsnaam worden beschouwd. De computer zal dus geen onderscheid tussen beide bestanden kunnen maken.



Je kunt soortnamen gebruiken om verschillende bestandsnamen voor de twee bestanden te maken.

PROGRAMMABESTANDEN EN GEGEVENSBESTANDEN

Wanneer een bestand een programma bevat, dan wordt dit een programmabestand genoemd. Een gegevensbestand is een bestand dat gegevens bevat. Gegevensbestanden worden verdeeld in twee soorten: volgorde- en direkt toegangelijke bestanden.
Deze onderverdeling wordt gemaakt aan de hand van de wijze waarop gegevens in- en uitgevoerd worden (het schrijven en lezen van gegevens).



In dit gedeelte zullen we alle bevelen behandelen die je kunt gebruiken voor programmabestanden en aangesloten apparatuur. Gegevensbestanden komen in de volgende paragraaf aan bod.

VERWERKEN VAN PROGRAMMABESTANDEN

De volgende bevelen dienen voor het opslaan van een BASIC programma in een bestand, het laden vanuit een bestand, en het kombineren van programma's.

CSAVE, CLOAD..................	uitsluitend voor cassetterecorder.
SAVE, LOAD, MERGE........	apparaat naar keuze.

Programma's opslaan, laden en het kombineren van programma's SAVE, LOAD, MERGE

Een programma dat met een CSAVE bevel opgeslagen werd, kan alleen met een CLOAD bevel geladen worden. Een programma dat met een SAVE bevel opgeslagen is, kan alleen met een LOAD bevel geladen worden. De bestandsnaam die gebruikt werd bij het opslaan van het programma, dient ook gebruikt te worden bij het laden van dit programma.
Het MERGE bevel laadt een programma en kombineert het met een programma dat al in het geheugen opgeslagen is. Je kunt alleen programma's die in de ASCII code zijn geschreven met elkaar kombineren.

Opslagbevel	Schrijfwijze	Laadbevel
CSAVE	Brugtaal	CLOAD
SAVE (cassetteband of geheugenschijf)	ASCII code	LOAD of MERGE
SAVE (diskette)	Brugtaal	LOAD
SAVE, A (diskette)	ASCII code	LOAD of MERGE

Overigens bestaat er wat betreft cassetteband wel verschil tussen de beide bevelen: een programma wordt met CSAVE opgeslagen in een zgn. brugtaal, terwijl het SAVE bevel het programma in ASCII code opslaat. Bij diskettes kun je alleen gebruik maken van het SAVE bevel. Door A aan het SAVE bevel toe te voegen, kun je bestanden in ASCII code op diskette opslaan. Bij het weglaten van deze toevoeging, wordt het programma in brugtaal op diskette opgeslagen.
Voor de geheugenschijf kan ook alleen het SAVE bevel worden gebruikt. Het is alleen maar mogelijk programma's in ASCII code in de geheugenschijf op te slaan.

Programmabestanden op cassetteband verwerken CSAVE, CLOAD
Een programma dat zich in het geheugen bevindt kun je op twee manieren op cassetteband opslaan:

```
CSAVE  "PROG"
```
in brugtaal opgeslagen met als bestandsnaam PROG— ①

```
SAVE  "CAS:PROG"
```
in ASCII code opgeslagen met als bestandsnaam PROG— ②

Om het programma dat hierboven in ① werd opgeslagen, te laden moet je het volgende uitvoeren:

```
CLOAD  "PROG"
```

Voor het laden van het programma dat hierboven in ② werd opgeslagen, voer je het onderstaande uit:

```
LOAD  "CAS:PROG"
```

Om het programma dat hierboven in ② werd opgeslagen, met een ander programma (dat zich al in het geheugen bevindt) te kombineren, moet je het volgende uitvoeren:

```
MERGE  "CAS:PROG"
```

Programmabestanden op diskette verwerken SAVE, LOAD
Je kunt een programma dat in het geheugen is opgeslagen, op twee manieren op diskette (diskette-eenheid A) opslaan:

```
SAVE  "A:PROG.BAS"
```
opgeslagen in brugtaal met de bestands-/soortnaam PROG.BAS— ①

```
SAVE  "A:PROG.ASC",A
```
opgeslagen in ASCII code met de bestands-/soortnaam PROG.BAS— ②

De programma's die hierboven bij ① en ② werden opgeslagen, kun je laden door het uitvoeren van:

```
LOAD  "A:PROG.BAS"

LOAD  "A:PROG.ASC"
```

Om een programma dat zich in het geheugen bevindt, te kombineren met het programma dat bij ② werd opgeslagen, moet je het volgende uitvoeren:

```
MERGE "A:PROG.ASC"
```

Wanneer je slechts een diskette-eenheid gebruikt, kun je de naam van de diskette-eenheid (A:) weglaten. Als bij gebruik van meerdere diskette-eenheden de naam van de diskette-eenheid weggelaten wordt, dan wordt de op dat moment in gebruik zijnde diskette-eenheid gekozen.

Programmabestanden in de geheugenschijf verwerken
Bij MSX2-BASIC kan een gedeelte van het interne geheugen gebruikt worden om programma's tijdelijk op te slaan. Gezien het feit dat het gebruik van dit gedeelte veel van het gebruik van een diskette weg heeft, wordt deze mogelijkheid aangeduid met de term geheugenschijffunktie. Het gedeelte van het geheugen waar normaal gesproken programma's worden opgeslagen, wordt het programmagebied genoemd. Programma's die in dit gebied opgeslagen zijn, kunnen met het LIST bevel op het scherm weergegeven worden en met het RUN bevel worden uitgevoerd. Wanneer je een programma dat in het programmagebied is opgeslagen in de geheugenschijf opslaat, dan kun je het gebruiken om hiermee een ander programma te schrijven. Echter, op het moment dat je de computer uitschakelt, wordt het programma dat zich in de geheugenschijf bevindt gewist.

Alvorens je de geheugenschijf kunt gebruiken, moet je deze eerst openen met het CALL MEMINI (geheugen openen) bevel.

CALL MEMINI [(afmetingen)]

Door de afmetingen in het CALL MEMINI bevel in te vullen, bepaal je het gedeelte van het geheugen dat door de geheugenschijf gebruikt kan worden. De afmetingen worden in bytes uitgedrukt. De afmetingen kunnen variëren van 1023 tot 32767 bytes. Bij het weglaten van de afmetingen in het bevel wordt er automatisch ingesteld op 32767 bytes. Bij het uitvoeren van het MEMINI bevel worden op het scherm de afmetingen van de geheugenschijf aangegeven.

```
CALL MEMINI
32000 bytes allocated
Ok
```

Als je eenmaal het MEMINI bevel hebt uitgevoerd, dan kun je een programma dat zich in het programmagebied bevindt, in de geheugenschijf opslaan door het volgende bevel te geven:

```
SAVE "MEM:PROG.BAS"
```
—— opgeslagen in ASCII code onder de bestands /soortnaam PROG.BAS

Om een programma uit de geheugenschijf in het programmagebied te laden, moet je het volgende bevel geven:

```
LOAD "MEM:PROG.BAS"
```

Voer het volgende bevel uit om een programma in de geheugenschijf met een programma in het geheugengebied te kombineren:

```
MERGE "MEM:PROG.BAS"
```

Laden en uitvoeren van een programma RUN
Het RUN bevel wordt gebruikt om een programma vanaf een diskette of de geheugenschijf te laden en onmiddellijk uit te voeren. RUN wordt gevolgd door de bestands- en soortnaam.

RUN "[apparaatnaam] bestandsnaam [.soortnaam]"

```
RUN "PROG.BAS"
```
—— "PROG.BAS" in diskette eenheid A wordt geladen en uitgevoerd

```
RUN "MEM:PROG.BAS"
```
—— "PROG.BAS" in de geheugenschijf wordt geladen en uitgevoerd

BEHEER VAN BESTANDEN

Weergeven van bestandsnamen FILES, CALL MFILES
Om de namen weer te geven van de bestanden die zich op een diskette bevinden, moet je het volgende bevel invoeren:

```
FILES
```

Om te kontroleren of een bepaald bestand op een diskette voorkomt, vul je de bestands/soortnaam in:

```
FILES "PROG.BAS"
```

Voer het volgende bevel uit om de namen van bestanden die zich in de geheugenschijf bevinden, weer te geven:

```
CALL MFILES
```

CALL MFILES

Na uitvoering van het CALL MFILES bevel wordt ook op het scherm aangegeven hoeveel bytes je nog kunt gebruiken in de geheugenschijf.

Wissen van bestanden KILL, CALL MKILL
Je kunt bestanden van een diskette wissen door gebruik te maken van het KILL bevel. Om bijvoorbeeld het bestand dat als bestandsnaam "TEST.DAT" heeft, te wissen zou je het volgende bevel uitvoeren:

```
KILL "TEST.DAT"
```

Het CALL MKILL bevel wordt gebruikt om bestanden die zich in de geheugenschijf bevinden, te wissen. Om het bestand genaamd "PROG.BAS" te wissen zou je het volgende uitvoeren:

```
CALL MKILL ("PROG.BAS")
```

CALL MKILL ("bestandsnaam [.soortnaam]")

Het veranderen van een bestandsnaam NAME, CALL MNAME

Met behulp van het NAME bevel kun je de bestands- en desgewenst de soortnaam van een bestand op een diskette veranderen.

NAME "[naam diskette-eenheid] oude bestandsnaam [.oude soortnaam]" AS "nieuwe bestandsnaam [.nieuwe soortnaam]"

CALL MNAME kun je gebruiken om de bestands- en soortnaam van een bestand dat zich in de geheugenschijf bevindt te veranderen.

CALL MNAME ("oude bestandsnaam [.oude soortnaam])" AS "(nieuwe bestandsnaam [.nieuwe soortnaam])"

```
NAME "OLD.BAS" AS "NEW.BAS"
```

———— "OLD.BAS" in diskette-eenheid A wordt veranderd in "NEW.BAS"

```
CALL MNAME ("OLD.BAS" AS "NEW.BAS")
```

———— "OLD.BAS" in de geheugenschijf wordt veranderd in "NEW.BAS"

PROGRAMMABESTAND MET AUTOMATISCHE START

Een programmabestand met een automatische start wordt automatisch van een diskette geladen en uitgevoerd, wanneer de computer ingeschakeld wordt of er op de RESET toets wordt gedrukt.
Vul het volgende als de bestands/soortnaam van het programma in:

AUTOEXEC.BAS

om bij het opslaan van het programma op diskette er een bestand met automatische start van te maken.
Je kunt per diskette maar één bestand met automatische start creëren.

GEBRUIK VAN VOLGORDE-BESTANDEN

- Wat is een volgorde-bestand?
- Gegevens schrijven in een volgorde-bestand
- Gegevens lezen uit een volgorde-bestand
- Gegevens toevoegen
- Lettertekens op een grafisch scherm schrijven
- Aantal tegelijk te openen bestanden

VOLGORDE-BESTANDEN EN DIREKT TOEGANKELIJKE BESTANDEN

Gegevensbestanden worden gebruikt voor gegevens die door een BASIC programma verwerkt zullen worden. Bijvoorbeeld in het geval van een telefoonboekprogramma wordt het programma zelf als programmabestand opgeslagen, maar de gegevens voor de namen en adressen die door het programma verwerkt worden, worden als gegevensbestand opgeslagen.

Er bestaan twee soorten gegevensbestanden:
1) Volgorde-bestanden waarin gegevens van het begin naar het eind van het bestand worden geschreven of gelezen;
2) Direkt toegankelijke bestanden waarin je op iedere gewenst punt kunt beginnen met schrijven of lezen.

De gebruiksmogelijkheden voor volgorde-bestanden en direkt toegankelijke bestanden staan hieronder beschreven:

Geschikt voor volgorde-bestanden	Geschikt voor direkt toegankelijke bestanden
cassetteband diskette afdrukker tekstscherm/grafisch scherm geheugenschijf	diskette

In dit gedeelte nemen we de diskette in diskette-eenheid A als voorbeeld voor het uitleggen van de volgorde-bestanden.
De volgende bevelen worden gebruikt voor het in- en uitvoeren van gegevens van en naar volgorde-bestanden.

OPEN	Opent een bestand.
PRINT # PRINT # USING	Voert gegevens uit naar een bestand.
INPUT # LINE INPUT #	Voert gegevens in vanuit een bestand.
CLOSE	Sluit een bestand.

GEGEVENS SCHRIJVEN IN EEN VOLGORDE-BESTAND OPEN FOR OUTPUT

Kort weergegeven is de gang van zaken voor uitvoer van gegevens naar een volgorde-bestand als volgt:
(1) Open een bestand met een OPEN bevel.
(2) Schrijf gegevens naar het bestand met een PRINT# bevel.
(3) Sluit het bestand met een CLOSE bevel.

De schrijfwijze van het OPEN bevel voor het uitvoeren van gegevens is:

OPEN "[apparaatnaam] [bestandsnaam [.soortnaam]]" FOR OUTPUT AS [#] bestandsnummer

Bij het verwerken hiervan wordt het uitvoeren van gegevens naar een nader omschreven apparaat met een nader omschreven bestandsnaam voorbereid. Wanneer daarna het bevel voor in- of uitvoer naar een bestand wordt gegeven slaat de computer eerst de gegevens op in zijn geheugen, om daarna over te gaan tot daadwerkelijke in- of uitvoer. Het in het geheugen voor de opslag van gegevens gereserveerde gebied wordt buffer genoemd. In MSX2-BASIC kunnen ten hoogste 16 buffers worden gereserveerd (7 is het maximale aantal bij een diskette). Het bestandsnummer in het OPEN bevel verwijst naar een van deze 16 buffers. De oorspronkelijke instelling is 1.

Nadat een bestand is geopend met het OPEN bevel wordt de uitvoer van gegevens in gang gezet door een PRINT# bevel. De schrijfwijze is:

PRINT# bestandsnummer, [uitdrukking] [scheidingsteken] [uitdrukking...]

Het bestandsnummer dient gelijk te zijn aan dat in het OPEN bevel. Bij het uitvoeren van gegevens naar een bestand met een PRINT# bevel wordt automatisch een terugkeercode (&H0D) en een regelopschuifcode (&H0A) aan de gegevens die in het PRINT# bevel zijn geschreven, toegevoegd. Deze twee codes worden bij het schrijven van nieuwe gegevens in het bestand als scheidingteken gebruikt.
Wanneer de gegevens bestaan uit meerdere rijen lettertekens die worden uitgevoerd met een PRINT# bevel dienen de rijen onderling gescheiden te worden door een komma tussen aanhalingstekens ",".
Het bevel zal dus in zijn geheel bijvoorbeeld zo luiden:

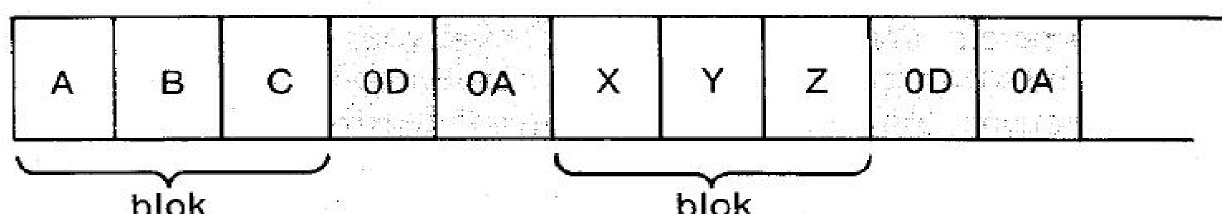
```
PRINT #1,A$;",";B$
```

De komma fungeert hier als scheidingsteken en de A$ en B$ rijen worden bij het lezen van de gegevens uit het bestand als aparte gegevens beschouwd. (Wanneer de gegevens uit getalswaarden bestaan worden ze automatisch onderling gescheiden.)

Iedere groep van gegevens die door een 0D,0A paar gescheiden wordt, wordt een blok genoemd. Wanneer bijvoorbeeld "ABC" aan de A$ variabele en "XYZ" aan de B$ variabele wordt toegewezen en

```
PRINT #1,A$
PRINT #1,B$
```

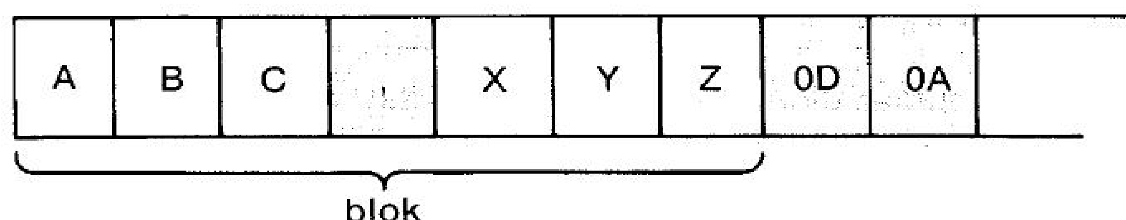
wordt uitgevoerd, dan worden de gegevens als volgt op de diskette geschreven:



Wanneer er een PRINT # bevel wordt uitgevoerd waarin beide variabelen geschreven zijn en deze van elkaar gescheiden worden door ",",, zoals in

```
PRINT #1,A$;",";B$
```

dan worden de gegevens als volgt geschreven:



In dit geval worden de gegevens ABC en XYZ in één blok geschreven, maar worden door de komma in twee eenheden gegevens verdeeld. Na het uitvoeren van gegevens dient een bestand gesloten te worden met een CLOSE bevel.

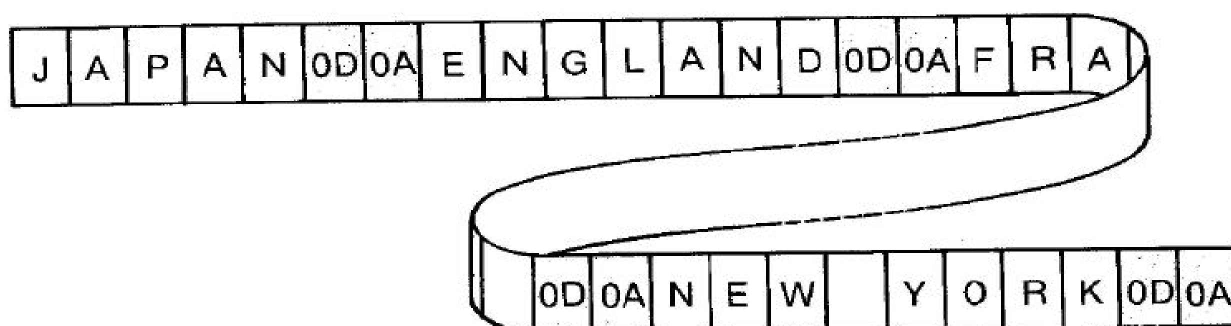
CLOSE [#] [bestandsnummer]

Aangezien door het sluiten van een bestand de relatie van het bestand en het nummer verbroken wordt, is het hierna mogelijk een ander bestand met hetzelfde bestandsnummer te openen.

Een programma waarbij er gegevens in een volgorde-bestand worden geschreven, is het volgende:

```
10 DIM A$(1,3)
20 OPEN "A:DATA.DAT" FOR OUTPUT AS #1
30 FOR L=0 TO 1
40 FOR M=0 TO 3
50 READ A$(L,M)
60 PRINT #1,A$(L,M)
70 NEXT M
80 NEXT L
90 CLOSE #1
100 END
110 DATA JAPAN,ENGLAND,FRANCE,U.S.A.
120 DATA TOKYO,LONDON,PARIS,NEW YORK
```

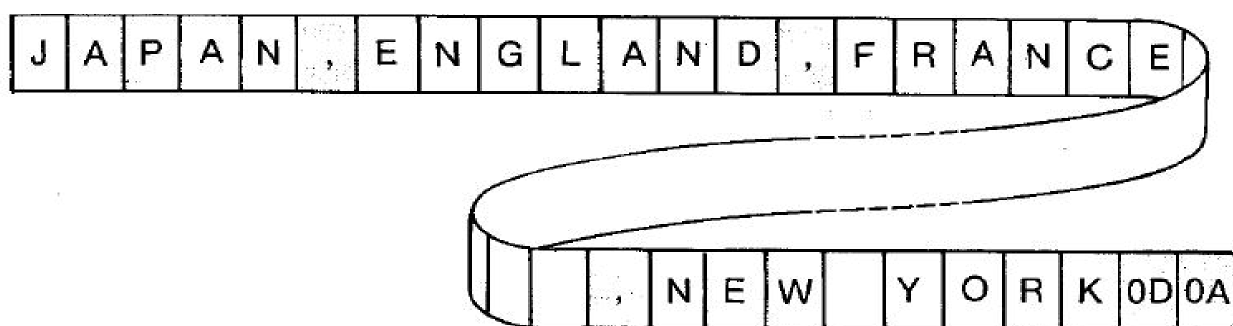
Bij uitvoering van programma wordt er een bestand met als naam "DATA.DAT" op de diskette in diskette-eenheid A gemaakt. Vervolgens worden de rij-gegevens "JAPAN", gevolgd door een 0D, 0A paar, "ENGLAND" enz. in die volgorde naar de diskette geschreven. De gegevens worden van elkaar gescheiden door een 0D, 0A paar.



Wanneer regel 60 gewijzigd wordt in:

```
PRINT #1,A$(L,M);",";
```

dan zullen de gegevens als volgt worden geschreven:


J A P A N , E N G L A N D , F R A N C E
, N E W Y O R K 0D 0A

GEGEVENS LEZEN UIT EEN VOLGORDE-BESTAND
OPEN FOR INPUT

Kort weergegeven is de gang van zaken voor invoeren (het lezen) van gegevens als volgt:
(1) Open een bestand met een OPEN bevel.
(2) Lees gegevens uit het bestand met een INPUT# of LINE INPUT# bevel.
(3) Sluit het bestand met het CLOSE bevel.

De schrijfwijze van het OPEN bevel voor het invoeren van gegevens is:

OPEN "[apparaatnaam] [bestandsnaam [.soortnaam]]" FOR INPUT AS [#] bestandsnummer

Bij het verwerken hiervan wordt het invoeren van gegevens vanuit een nader omschreven bestand voorbereid. Alleen het cijfer 1 kan als bestandsnummer gebruikt worden, tenzij je een andere instelling hebt gemaakt met het MAXFILES bevel (zie hiervoor blz. 255).
Nadat een bestand is geopend met het OPEN bevel wordt de invoer van gegevens in gang gezet door een INPUT# bevel.

INPUT[#] bestandsnummer, variabele

Het INPUT# bevel wijst een groep gegevens toe aan de variabele. De gegevens die worden gelezen met een INPUT# bevel worden verwerkt op de wijze als aangegeven in onderstaande tabel.

	Bij numerieke gegevens	Bij rijen lettertekens
Spatie, terugkeercode, regelopschuifcode voor de gegevens	Worden genegeerd	Worden genegeerd
Scheidingstekens of leestekens tussen de gegevens	Spatie, komma, terugkeercode, regelopschuifcode	Komma, terugkeercode, regelopschuifcode. Voor invoer van 255 lettertekens.
Aanhalingstekens die de gegevens omsluiten (" ")	—	Invoer tussen " " wordt als één gegeven beschouwd.

Het LINE INPUT# bevel dient alleen voor het uitlezen van een rij letterteken-gegevens die worden ingevoerd met als enige scheidingsteken de terugkeercode.
Na het beëindigen van de invoer van gegevens wordt het bestand gesloten met een CLOSE bevel om de relatie tussen het bestand en het bestandsnummer te verbreken.

Laten we een programma schrijven dat de gegevens uit het eerder gemaakte "DATA.DAT" bestand leest en deze op het scherm weergeeft.

```
10  DIM A$(1,3)
20  OPEN "A:DATA.DAT" FOR INPUT AS #1
30  FOR L=0 TO 1
40  FOR M=0 TO 3
50  INPUT #1,A$(L,M)
60  NEXT M
70  NEXT L
80  CLOSE #1
90  FOR M=0 TO 3
100  PRINT A$(0,M),A$(1,M)
110  NEXT M
```

Door het INPUT# bevel in regel 50 worden de gegevens alle toegewezen aan de A$ lijstvariabele. Door de regels 90 t/m 110 worden de gegevens op het scherm weergegeven.

Wat gebeurt er als we proberen met dit programma de gegevens in het bestand "DATA.DAT" in te voeren?

```
10  OPEN "A:DATA.DAT" FOR INPUT AS #1
20  INPUT #1,A$
30  PRINT A$
40  GOTO 20
```

JAPAN, ENGLAND ... en alle andere gegevens worden toegekend aan de A$ variabele en vervolgens op het scherm weergegeven. Na het invoeren van het laatste gegeven, NEW YORK, blijft het programma proberen meer gegevens in te voeren. Wanneer dit zich voordoet en het einde van een bestand wordt a.h.w. overschreden, dan treedt er een

Input past end

fout op. Dit kunt je voorkomen met de EOF (voor "end of file") funktie.

EOF (bestandsnummer)

```
10 OPEN "A:DATA.DAT" FOR INPUT AS #1
15 IF EOF(1)=-1 THEN GOTO 50
20 INPUT #1,A$
30 PRINT A$
40 GOTO 15
50 CLOSE #1
```

De EOF(bestandsnummer) funktie geeft de waarde -1 wanneer het laatste gegeven van een bestand uitgelezen is. In dit programma kontroleert de funktie telkens voor het invoeren van een gegeven of er inderdaad nog wel gegevens in het bestand over zijn.

GEGEVENS TOEVOEGEN OPEN FOR APPEND

Je kunt alleen gegevens toevoegen aan een bestand dat je eerder hebt gemaakt, wanneer het een volgorde-bestand betreft en dit bestand zich op diskette of in de geheugenschijf bevindt.
Om gegevens toe te voegen moet je het bestand eerst openen met het OPEN bevel.

OPEN "[apparaatnaam] [bestandsnaam [.soortnaam]]" FOR APPEND AS [#] bestandsnummer

De volgende drie programma's schrijven, lezen en voegen gegevens toe aan een bestand op een diskette.

Gegevens schrijven

```
10 OPEN "A:TEST.DAT" FOR OUTPUT AS #1
20 FOR L=1 TO 3
30 READ A$
40 PRINT #1,A$
50 NEXT L
60 CLOSE #1
70 END
80 DATA JAPAN,ENGLAND,FRANCE
```

Dit programma creëert een bestand op de diskette in diskette-eenheid A onder de naam "TEST.DAT". Vervolgens worden de drie gegevens
—JAPAN, ENGLAND, FRANCE—in het bestand geschreven.

Gegevens lezen

```
10 OPEN "A:TEST.DAT" FOR INPUT AS #1
20 IF EOF(1)=-1 GOTO 60
30 INPUT #1,A$
40 PRINT A$
50 GOTO 20
60 CLOSE #1
70 END
```

LETTERTEKENS OP HET GRAFISCHE SCHERM SCHRIJVEN

Na het kiezen van een van de grafische schermen is het niet meer mogelijk lettertekens op het scherm te brengen met een PRINT bevel. Om desondanks lettertekens op het grafische scherm weer te geven bedienen we ons van een methode waarbij het grafische scherm wordt opgevat als een bestandsverwerkend apparaat, en de lettertekens, die we willen afbeelden, als een volgorde-bestand met gegevens.

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PRINT #1,"Hoe gaat het?"
40 GOTO 40
```

Bij het verwerken van dit programma wordt het grafische scherm gekozen en "Hoe gaat het ermee?" weergegeven.

Voor het bepalen van de plaats waar de letters op het scherm komen is het nodig voor het PRINT # bevel eerst een grafisch bevel te geven. Daarna wordt de laatst genoemde positie hiervan (een van de 256 horizontaal bij 192 vertikale stippen) gebruikt als de linkerbovenhoek van een 8×6 blok van stippen dat plaats biedt aan het eerste letterteken van de met PRINT uitgevoerde rij.

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
25 PRESET (100,50)
30 PRINT #1,"Hoe gaat het?"
40 GOTO 40
```

In dit programma wordt de positie (100,50) van het PRESET bevel in regel 25 gebruikt als de linkerbovenhoek van de rij lettertekens die wordt uitgevoerd in regel 30.

De gegevens in het schrijfprogramma worden gelezen, aan de A$ variabele toegewezen en op het scherm weergegeven.

Gegevens toevoegen

```
10 OPEN "A:TEST.DAT" FOR APPEND AS #1
20 FOR L=1 TO 2
30 READ A$
40 PRINT #1,A$
50 NEXT L
60 CLOSE #1
70 END
80 DATA U.S.A.,CHINA
```

Het "TEST.DAT" bestand dat in regel 10 van dit programma wordt geopend, is het programma waar we eerder met het schrijfprogramma al drie gegevens in hebben gezet. Omdat er echter in het OPEN bevel FOR APPEND is ingevuld, worden de gegevens die geschreven zijn door het PRINT# bevel in regel 40, achter de drie eerder genoemde gegevens geplaatst. Bij uitvoer van dit programma zal het "TEST.DAT" bestand dan ook vijf gegevens bevatten: JAPAN, ENGLAND, FRANCE, U.S.A. en CHINA.
Als je in plaats van FOR APPEND in het bevel FOR OUTPUT had ingevuld, dan waren de nieuwe gegevens aan het begin van het bestand geschreven en zouden de bestaande drie gegevens gewist zijn.

AANTAL TEGELIJK TE OPENEN BESTANDEN
MAXFILES

Bij het inschakelen van MSX2-BASIC is het niet mogelijk andere bestandsnummers dan bestandsnummer 1 in te voeren. Met andere woorden, er kan in een programma maar één bestand tegelijk geopend zijn. Voor het openen van meerdere bestanden tegelijk moet het gewenste aantal eerst worden aangegeven met het bevel:

MAXFILES = uitdrukking

Als je bijvoorbeeld

```
MAXFILES=5
```

invult, dan kunnen er in totaal 6 bestanden (met de bestandsnummers 0 t/m 5) tegelijk geopend worden.
Het grootste aantal bestanden dat op deze wijze gelijk te openen is bedraagt 16 (van bestandsnummer 0 t/m 15). (Bij gebruik van een diskette beperkt zich dit tot bestandsnummer 0 t/m 6.)
Overigens is een bestand met het bestandsnummer 0 speciaal bestemd voor gebruik met de CSAVE, CLOAD, CLOAD?, LOAD en SAVE bevelen, zodat na het bevel

```
MAXFILES=0
```

voor in- en uitvoer alleen de bevelen CSAVE, CLOAD, CLOAD?, LOAD en SAVE beschikbaar zijn.
Het MAXFILES bevel moet je aan het begin van een programma of rechtstreeks, buiten een programma, geven.

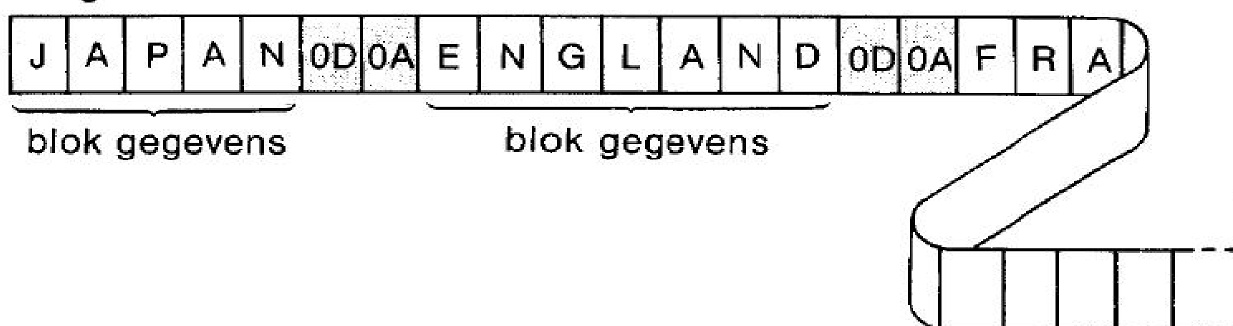
GEBRUIK VAN DIREKT TOEGANKELIJKE BESTANDEN

- Wat is een direkt toegankelijk bestand?
- Gegevens schrijven in een direkt toegankelijk bestand
- Gegevens lezen uit een direkt toegankelijk bestand

WAT IS EEN DIREKT TOEGANKELIJK BESTAND?

Vergeleken met een volgorde-bestand
Een direkt toegankelijk bestand is alleen in kombinatie met een diskette te gebruiken. In het onderstaande schema kun je zien hoe het schrijven van gegevens in zijn werk gaat bij een volgorde-bestand en hoe bij een direkt toegankelijk bestand.

volgorde-bestand



direkt toegankelijk bestand

blok 1 JAPAN TOKYO
blok 2 ENGLAND LONDON
blok 3 FRANCE PARIS
blok 4

22 bytes

Zoals je in de bovenstaande afbeelding kunt zien worden de gegevens in een volgorde-bestand geschreven alsof ze op een band of een rol papier komen te staan. Een blok is de eenheid voor het een keer schrijven in het bestand, en de lengte van een blok is afhankelijk van de hoeveelheid gegevens.
Een direkt toegankelijk bestand lijkt daarentegen meer op een aantal stroken papier die allemaal op dezelfde lengte zijn afgeknipt; elk van de stroken is een afzonderlijk blok, apart genummerd 1, 2, 3 enz. In de bovenstaande afbeelding zijn de stroken allemaal op een zodanige lengte afgeknipt dat ze 22 lettertekens kunnen bevatten. We zeggen dat het formaat van elk blok 22 "bytes" bedraagt. Een byte staat voor een letterteken, of dit nu een letter van het alfabet is, een cijfer, een komma of zelfs een spatie. Al deze tekens worden als 1 byte geteld. Aangezien de blokken van een direkt toegankelijk bestand net afzonderlijke stroken papier zijn is het desgewenst mogelijk één enkel blok te kiezen om er gegevens in te schrijven of uit te lezen. Als we bijvoorbeeld blok 3 kiezen, dan kunnen we de gegevens "FRANCE PARIS" eruit halen. Of we kunnen blok nummer 10 kiezen en daarnaartoe overspringen om er nieuwe gegevens in te schrijven. Nog iets wat het opmerken waard is, is dat zelfs binnen elk afzonderlijk blok van een direkt toegankelijk bestand de plaatsen waar de gegevens worden geschreven keurig netjes op een rijtje staan.

Bevelen en instrukties voor direkt toegankelijke bestanden

De volgende bevelen dienen voor de in- en uitvoer van gegevens naar en vanuit een direkt toegankelijk bestand.

OPEN .... bestand openen
FIELD .... formaat van een blok bepalen
LSET, RSET .... gegevens in een blok schrijven
PUT .... een blok naar bestand uitvoeren
GET .... een blok vanuit bestand invoeren
CLOSE .... bestand sluiten

GEGEVENS SCHRIJVEN IN EEN DIREKT TOEGANKELIJK BESTAND

Voor het schrijven van gegevens in een direkt toegankelijk bestand ga je als volgt te werk.
(1) Open het bestand met het OPEN bevel.
(2) Kies het formaat van een blok met de FIELD instruktie.
(3) Schrijf gegevens in een blok met het LSET of RSET bevel.
(4) Voer de gegevens uit naar het bestand met het PUT bevel.
(5) Sluit het bestand.

Elk van de bovenstaande stappen wordt in dit hoofdstuk nader uiteengezet.

(1) Open het bestand met het OPEN bevel
Voor het inschrijven of uitlezen van gegevens kan een bestand geopend worden met het OPEN bevel. De schrijfwijze van dit bevel luidt:

OPEN "[apparaatnaam] bestandsnaam [.soortnaam]" AS[#] bestandsnummer [LEN = bloklengte]

De apparaatnaam is altijd gelijk aan de diskette-eenheidsnaam, aangezien een direkt toegankelijk bestand alleen gebruikt kan worden in kombinatie met een diskette.

Na verwerking van het OPEN bevel is op de aangegeven diskette een bestand met de aangegeven bestandsnaam en soortnaam in gereedheid voor de uitvoer van gegevens. Net als bij volgorde-bestanden kan voor het bestandsnummer over het algemeen een getal van 0 tot 6 worden gekozen, maar in de beginstand kan je alleen het nummer "1" invoeren.
Het voordeel van een direkt toegankelijk bestand in vergelijking met volgorde-bestanden is dat je de gegevens kunt schrijven in, of lezen uit elk gewenst gedeelte van het bestand. De kleinste eenheid die geschreven of gelezen kan worden heet een blok (record). De afmetingen van een blok, uitgedrukt in bytes, worden vastgesteld door invullen van de "bloklengte" in het OPEN bevel. Het formaat kan ingesteld worden op elke gewenste lengte van 1 t/m 256 bytes. Bij weglaten van de bloklengte in het bevel wordt het formaat automatisch ingesteld op 256 bytes.

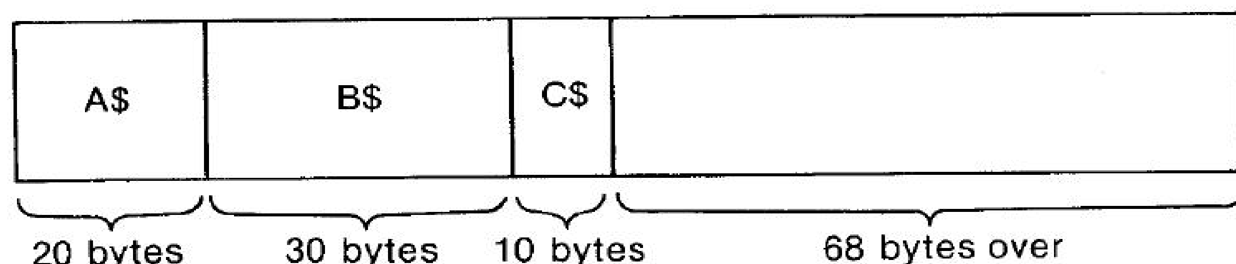
(2) Kies het formaat van een blok met de FIELD instruktie.

FIELD[#] bestandsnummer, aantal lettertekens AS rij-variabele [,aantal lettertekens AS rij-variabele]...

De FIELD instruktie dient voor het kiezen van de variabelen die je voor lezen en schrijven van gegevens wilt gebruiken en voor het bepalen van het aantal lettertekens dat elk van de variabelen in een blok kan bevatten.
Alle gegevens die in een direkt toegankelijk bestand gebruikt worden moeten lettertekenrij-gegevens zijn.

```
FIELD #1,20 AS A$,30 AS B$,10 AS C$
```

In het bovenstaande voorbeeld wordt in bestand #1 een blok voor in- en uitvoer in stukjes verdeeld, waarbij 20 bytes aan de rij-variabele A$ worden toegewezen, 30 bytes aan rij-variabele B$ en 10 bytes aan C$. De totale lengte van deze rij-variabelen bedraagt dus 60 bytes, zodat van tevoren voor de bloklengte in het voorgaande OPEN bevel 60 bytes ingevuld moet worden. Als de bloklengte van tevoren bijvoorbeeld is ingesteld op 128 bytes, dan splitst de FIELD instruktie het blok als volgt op:



Zoals je ziet worden in dit voorbeeld voor de resterende 68 bytes geen gegevens in- of uitgevoerd, hetgeen betekent dat er nodeloos ruimte op de diskette verspild wordt. Daarom verdient het de voorkeur de bloklengte in het OPEN bevel op slechts 60 bytes te bepalen.

(3) Schrijf gegevens in een blok met het LSET of RSET bevel.
Nadat een blok aldus is opgesplitst, en er variabelen aan zijn toegewezen met de FIELD instruktie kunnen de uitgevoerde gegevens op hun plaats in het blok gezet worden. Het LSET bevel schrijft de gegevens vanaf de linkerkant in het blok, terwijl het RSET bevel zorgt dat de gegevens geheel aan de rechterkant van het blok komen te staan.

LSET rij-variabele = lettertekenrij
RSET rij-variabele = lettertekenrij

De rij-variabelen in deze bevelen zijn dezelfde als eerder met het FIELD bevel aan het blok zijn toegewezen. De rijtjes lettertekens zijn de gegevens die je met behulp van de rij-variabelen in het bestand wilt schrijven.
Door invoeren van het bevel

```
LSET A$=X$
```

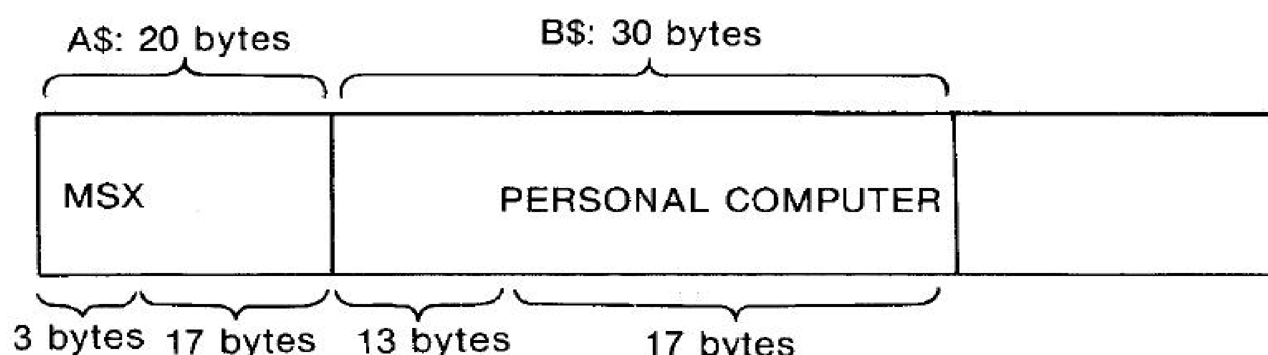
wordt de X$ lettertekenrij (de eerste rij gegevens) als waarde aan de A$ variabele toegewezen en vanaf de linkerkant in het blok geschreven.
Met het bevel

```
RSET B$=Y$
```

wordt de Y$ lettertekenrij (de volgende rij gegevens) als waarde aan de B$ variabele toegewezen en vanaf de rechterkant in het blok geschreven.
Als geldt dat

```
X$="MSX"
Y$="PERSONAL COMPUTER"
```

dan worden deze gegevens op de hieronder aangegeven wijze in het blok gezet:
(Opmerking: Op dit punt wordt het blok alleen nog maar in de geheugenbuffer gezet. Om de gegevens in het bestand op de diskette te schrijven volgt later het PUT bevel. Dit is een voorbeeld van de ordelijke manier waarop en direkt toegankelijk bestand met gegevens omgaat.)



Als de gegevens meer lettertekens beslaan dan er bytes aan de variabelen in het blok zijn toegewezen, dan worden de laatste lettertekens (aan het rechter uiteinde) genegeerd.

Nu kunnen tekstgegevens in het bestand geschreven worden. Aangezien echter blokken alleen lettertekenrij-variabelen kunnen bevatten, moeten eventueel aanwezige numerieke gegevens voor het schrijven in het blok eerst omgezet worden in lettertekenrij-gegevens. Hiervoor kunnen de funkties MKI$ (maak geheel getal dollar), MKS$ (maak enkele precisie dollar) en MKD$ (maak dubbele precisie dollar) in de LSET en RSET bevelen toegepast worden. Deze drie funkties worden uitsluitend gebruikt voor direkt toegankelijke bestanden, waarbij elk van de drie dient voor het omzetten van een bepaald type getalswaarde in een rij-waarde (ofwel een rijtje lettertekens).

LSET (of RSET) rij-variabele = MKI$ (gehele getallen-gegevens)
LSET (of RSET) rij-variabele = MKS$ (enkele precisie-gegevens)
LSET (of RSET) rij-variabele = MKD$ (dubbele precisie-gegevens)

```
LSET P$ = MKI$(A%)
LSET Q$ = MKS$(B!)
LSET R$ = MKD$(C#)
```

Nadat de numerieke gegevens met deze funkties zijn omgezet in lettertekenrijtjes is de lengte in bytes hiervan gelijk aan 2 bytes voor gehele getallen, 4 bytes voor enkele-precisie waarden en 8 bytes voor dubbele-precisie waarden. Bij het verdelen van blokken met het FIELD bevel moeten voor numerieke gegevens minimaal deze lengten worden aangehouden.
In het bovenstaande voorbeeld is variabele P$ voor gehele getallen, Q$ voor enkele-precisie gegevens en R$ voor dubbele-precisie gegevens. Deze variabelen zouden als volgt in een FIELD instruktie ingesteld moeten worden:

```
FIELD #1,2 AS P$,4 AS Q$,8 AS R$
```

(4) Voer de gegevens uit naar het bestand met het PUT bevel.
Als de gegevens eenmaal in het blok in de geheugenbuffer gezet zijn, kun je vervolgens het PUT bevel geven om de gegevens in het bestand op de diskette te schrijven.

PUT[#] bestandsnummer [,bloknummer]

Het PUT bevel dient om de gegevens die zich in een blok bevinden uit te schrijven naar een voor het blok gekozen plaats in het bestand dat wordt aangegeven met het bestandsnummer. Het bloknummer dient om de gegevens later uit het direkt toegankelijk bestand te kunnen lezen.
Als je bij het schrijven van het PUT bevel het bloknummer weglaat, dan wordt hiervoor automatisch het nummer 1 gekozen, mits er nog geen eerder PUT of GET bevel verwerkt is. Als dit wel het geval is, dan wordt voor het verkrijgen van het nieuwe bloknummer automatisch bij het bloknummer van het voorgaande PUT of GET bevel 1 opgeteld.

(5) Sluit het bestand.
Om een bestand te sluiten geef je het
CLOSE[#] [bestandsnummer]
bevel, waardoor het aan het bestand toegewezen nummer vervalt en desgewenst weer voor een ander bestand beschikbaar komt.

Het onderstaande programma is een voorbeeld van een volledig programma voor het schrijven van gegevens in een direkt toegankelijk bestand.

```
10 OPEN "A:TELNO.DAT" AS #1
20 FIELD #1,2 AS ID$,12 AS NAM$,11 AS NO
$
30 FOR R=1 TO 3
40 READ A%,B$,C$
50 LSET ID$=MKI$(A%)
60 LSET NAM$=B$
70 RSET NO$=C$
80 PUT #1,R
90 NEXT R
100 CLOSE #1
110 DATA 1,TOM,111-2222
120 DATA 2,SUSAN,333-4444
130 DATA 3,JOAN,555-6666
```

A% in regel 40 en 50 is een numerieke variabele voor gehele getallen. (Voor het bepalen van de soort numerieke variabele zijn er de definitietekens % voor geheel-getal variabelen, ! voor enkele-precisie variabelen en # voor dubbele-precisie variabelen.)

Bij verwerken van het bovenstaande voorbeeldprogramma wordt op de diskette in diskette-eenheid A een bestand "TELNO.DAT" geopend. De totale lengte van 1 blok voor in- en uitvoer van gegevens is ingesteld op 25 bytes, waarvan 2 bytes zijn toegekend aan de rij-variabele ID$, 12 bytes aan de variabele NAM$ en 11 bytes aan NO$. Door het drie maal herhalen van de FOR-NEXT lus die begint bij regel 30 worden de volgende gegevens in het bestand geschreven.

	ID$: 2 bytes	NAM$: 12 bytes	NO$: 11 bytes
blok 1	(1)	TOM	111-2222
blok 2	(2)	SUSAN	333-4444
blok 3	(3)	JOAN	555-6666

De geheel-getal gegevens 1, 2 en 3 in deze kolom worden omgezet in lettertekenrij-gegevens, en gezien het soort gegevens worden ze elk als 2 bytes geschreven.

GEGEVENS LEZEN UIT EEN DIREKT TOEGANKELIJK BESTAND

De werkwijze voor het lezen van gegevens uit een direkt toegankelijk bestand is ongeveer gelijk aan de hierboven beschreven stappen voor het inschrijven.
(1) Open het bestand met het OPEN bevel.
(2) Kies het formaat van een blok met de FIELD instruktie.
(3) Lees de gegevens uit een blok met behulp van het GET bevel.
(4) Sluit het bestand.

De stappen (1) en (2) zijn gelijk aan die voor het schrijven van de gegevens in het bestand. De gegevens worden daadwerkelijk gelezen door het GET bevel in stap (3).

GET[#] bestandsnummer, [,bloknummer]

Het GET bevel leest de gegevens uit een blok dat is aangegeven met het bloknummer en wijst ze als waarden toe aan elk van de variabelen die zijn ingesteld met de FIELD instruktie.
De numerieke gegevens waren voor het inschrijven in het bestand omgezet in lettertekenrij-gegevens, en bij het lezen worden ze dus toegewezen aan de rij-variabelen in de FIELD instruktie. Om de rij-gegevens op het scherm weer te geven en ze verder te verwerken is het nodig ze weer terug te veranderen in numerieke gegevens, hun oorspronkelijke vorm. Hiervoor dienen de CVI, CVS en CVD funkties, waarvan de werking dus precies het omgekeerde is van de funkties MKI$, MKS$ en MKD$. Eerst worden bijvoorbeeld gegevens die oorspronkelijk numeriek waren, van het geheel-getal type, door het GET bevel gelezen en aan de rij-variabele P$ toegewezen. Dan worden deze gegevens weer omgezet in geheel-getal gegevens en toegewezen aan de geheel-getal variabele A% door

```
A%=CVI(P$)
```

CVI(P$) kan ook in een losstaand PRINT bevel geschreven worden en rechtstreeks verwerkt worden om de gegevens af te beelden.

```
PRINT CVI(P$)
```

CVI (van het Engelse "ConVert to Integer") dient voor het omzetten van rij-gegevens in geheel-getal gegevens; CVS (van ConVert to Single precision) voor het omzetten van rij-gegevens in enkele-precisie gegevens; en CVD (van ConVert to Double precision) voor het omzetten van rij-gegevens in dubbele-precisie gegevens.

numerieke variabele (gehele getallen) = CVI (rij-gegevens)
numerieke variabele (enkele precisie) = CVS (rij-gegevens)
numerieke variabele (dubbele precisie) = CVD (rij-gegevens)

Nadat de gegevens gelezen zijn wordt het bestand weer gesloten met het CLOSE bevel.

Laten we nu dan maar een programma schrijven voor het lezen van de gegevens uit het direkt toegankelijk bestand dat we in het voorgaande programma hebben gemaakt, en voor het weergeven van de gegevens op het scherm.

```
10 SCREEN 0:WIDTH 40
20 OPEN "A:TELNO.DAT" AS #1
30 FIELD #1,2 AS ID$,12 AS NAM$,11 AS NO
$
40 INPUT "Bloknummer  ";N
50 IF N=0 THEN GOTO 110
60 GET #1,N
70 PRINT "ID NO.";CVI(ID$);"   ";
80 PRINT NAM$;NO$
90 PRINT
100 GOTO 40
110 CLOSE #1
```

Als alles naar behoren verloopt, worden de gegevens na uitvoeren van het programma op het scherm gezet zoals aangegeven in onderstaande afbeelding:

```
RUN
Bloknummer? 1
ID NO.1     TOM                 111-2222

Bloknummer? 2
ID NO.2     SUSAN               333-4444

Bloknummer? 0
Ok
```

Hoofdstuk 10
Subroutines in Machinetaal

SCHRIJVEN EN UITVOEREN VAN MACHINETAAL-SUBROUTINES

- Instellen van begin- en eindadres—CLEAR, DEFUSR
- Schrijven van een machinetaal-subroutine—POKE
- Oproepen van een machinetaal-subroutine—USR
- Opslaan en laden van een subroutine—BSAVE, BLOAD

MACHINETAAL-SUBROUTINES

MSX-BASIC biedt de mogelijkheid een programma te schrijven in de machinetaal van de Z-80A (de centrale verwerkingseenheid van de MSX persoonlijke computer). Er wordt dan vanuit BASIC overgeschakeld naar de machinetaal voor het sturen van de computer, en nadat het programma in machinetaal geheel verwerkt is kan het resultaat worden toegewezen aan een variabele die in BASIC gedefinieerd is. Er kunnen in totaal 10 subroutines in machinetaal geschreven en gebruikt worden. Per subroutine kan slechts een waarde toegewezen worden.

GEBIED EN BEGINADRES VAN EEN SUBROUTINE BEPALEN CLEAR, DEFUSR

Voor het schrijven van de machinetaal subroutine **moet eerst met behulp van een CLEAR bevel een gebied in het geheugen hiervoor apart gezet worden. Het CLEAR bevel dient voor het bepalen van het hoogst genummerde adres van het geheugen-gebied dat voor BASIC programma's gebruikt wordt**. Het gebied volgend op dit adres kan daarna gebruikt worden voor het schrijven van een subroutine in machinetaal.

CLEAR [afmetingen tekstgebied] [,hoogste geheugenadres]

Het bevel

```
CLEAR 300,&HCFFF
```

Stelt het adres &HCFFF (decimaal adres 53247) in als hoogste adres van het geheugen-gebied voor BASIC programma's. Het gebied hierna, te beginnen met adres &HD000 (decimaal adres 53248) kan dan gebruikt worden voor het schrijven van een machinetaal-subroutine. In het bovenstaande CLEAR bevel is de grootte van het tekstgebied ingesteld op 300 bytes. Bij weglaten van deze instelling geldt voor de afmetingen van het tekstgebied de oorspronkelijke waarde van 400 bytes.

Vervolgens kan je het beginadres van de subroutine met een DEFUSR bevel bepalen.

DEFUSR[X] = beginadres

X kan een geheel getal zijn van 0 tot 9. Er kunnen 10 subroutines met hun beginadressen als USR funkties gedefinieerd worden.

```
DEFUSR0=&HD000
```

In dit DEFUSR bevel wordt een machinetaal subroutine vanaf adres &HD000 (decimaal adres 53248) gedefinieerd als de USR 0 funktie.

SCHRIJVEN VAN EEN MACHINETAAL-SUBROUTINE POKE

Om een machinetaal-subroutine in het geheugen te schrijven gebruiken we het POKE bevel.

POKE adres, uitdrukking

Het POKE bevel schrijft één byte gegevens naar het aangegeven adres in het geheugen.
Het volgende programma schrijft de hexadecimale getallen 21, 3C, F0 en C9 in het geheugen, te beginnen bij het adres &HD000.

```
100 AD=&HD000
110 READ M$:IF M$="END" THEN END
120 POKE AD,VAL("&H"+M$)
130 AD=AD+1:GOTO 110
140 DATA 21,3C,F0,C9
150 DATA END
```

Hierna is het voldoende om de bevelen van de subroutine in machinetaal (de instrukties voor de Z-80 processor) in de DATA instruktie van regel 140 te schrijven. De RET instruktie zorgt dat na verwerking van de machinetaal-subroutine de besturing van de computer weer wordt overgenomen door het BASIC hoofdprogramma.

OPROEPEN VAN EEN MACHINETAAL-SUBROUTINE USR

Het overschakelen naar een subroutine in machinetaal voor de besturing van de computer wordt aangeduid met de term "oproepen van een machinetaal-subroutine".

Voor het oproepen van een machinetaal-subroutine gebruiken we de USR funktie.

USR[X](I)

X staat voor het nummer van de USR funktie, zoals vastgelegd in het DEFUSR bevel. I is de waarde (getalswaarde of lettertekenrij-waarde) die wordt toegewezen aan de machinetaal-subroutine.
De subroutine die is gedefinieerd met DRFUSR 0 wordt bijvoorbeeld als volgt opgeroepen:

```
X=USR0(Y)
```

Door deze instruktie wordt de verwerking van de machinetaal-subroutine gestart. Nadat de verwerking beëindigd is wordt het eindresultaat aan de gekozen variabele X toegekend, waarna zonder onderbreking het BASIC programma weer hervat wordt.
Bij het oproepen van de machinetaal-subroutine, oftewel het overschakelen vanuit BASIC naar een machinetaal-subroutine, wordt de waarde van de Y variabele, de parameter van de USR funktie, opgeslagen op één van de hieronder gegeven geheugenadressen. Welk adres is afhankelijk van het soort variabele dat Y is, en tegelijk wordt deze laatste informatie, het soort of type, in register A vastgelegd. Het beginadres, waar de waarde van Y is opgeslagen, wordt ingevoerd in de HL registers.

Soort of type variabele Y	Gegeven in A* register	HL registers adres-aanduiding	Adres waar de waarde van Y is vastgelegd
Geheel getal	2	&HF7F6	&HF7F8—&HF7F9
Enkele precisie getalswaarde	4		&HF7F6—&HF7F9
Dubbele precisie getalswaarde	8		&HF7F6—&HF7FD

* Dezelfde waarde wordt vastgelegd op geheugenadres &HF663.

Wanneer Y een rij-variabele is zijn de verschillende gegevens als volgt:

Gegeven in register A	Gegeven in de registers DE	Beschrijving rij lettertekens
3	Beginadres beschrijving rij letter- tekens	Eerste byte: lengte van de rij lettertekens Tweede en derde byte: beginadres van het gebied waar de rij lettertekens is opgeslagen

Na het volledig verwerken van de machinetaal subroutine wordt het eindresultaat aan variabele X toegewezen, door de registers en het geheugen tijdens het beëindigen in te stellen, zoals in de volgende tabel aangegeven:

Soort of type eindresultaat	&HF663 geheugen- adres	DE regis- ters	HL regis- ters	Opslagadres eindresultaat
Geheel getal	2		&HF7F6	&HF7F8—&HF7F9
Enkele precisie getalswaarde	4		&HF7F6	&HF7F6—&HF7F9
Dubbele precisie getalswaarde	8		&HF7F6	&HF7F6—&HF7FD
Rij lettertekens	3	Begin- adres beschrij- ving rij letter- tekens		Beginadres op- slaggebied aan- gegeven door tweede en derde byte van rij- beschrijving.

Het gegenereerde getal, dat moet worden teruggeven aan BASIC, wordt opgeslagen in de adressen &HF7F8 en &HF7F9. Aangezien het hier een geheel getal betreft wordt voor de soort waarde een 2 in het adres &HF663 ingevoerd. Het volgende BASIC programma schrijft de machinetaal-subroutine in het geheugen, zorgt voor het oproepen ervan, en gebruikt de teruggegeven waarden (gehele getallen tussen 0 en 255) voor het afbeelden van één tot zes ogen op de dobbelsteen.

```
10 CLEAR 300,&HCFFF          — bepaalt het hoogste geheugenadres
20 SET BEEP 1                  van het BASIC programmagebied
30 '--                         bepaalt het beginadres
40 AD=&HD000:DEFUSR0=AD      — van de subroutine op
50 GOSUB 430                   &HD000
60 ' --
70 SCREEN 5
80 OPEN "GRP:" FOR OUTPUT AS #1
90 SET PAGE 1,1:CLS
100 R=5:RESTORE 360
110 FOR L=0 TO 5                                  tekent de
120 XC=(L MOD 3)*80                               ogen van de
130 YC=(L \ 3)*100                                dobbelsteen
140 LINE (XC,YC)-STEP(50,50),15,BF                (van een tot
150 FOR M=0 TO L                                  zes) op
160 READ X,Y                                      pagina 1
170 CIRCLE (XC+X,YC+Y),R,8
180 PAINT (XC+X,YC+Y),8,8
190 NEXT M
200 NEXT L
210 ' --
220 SET PAGE 0,0                 zorgt dat de subroutine
230 PRESET (70,150)              wordt opgeroepen
240 PRINT #1,"Druk een toets in" na indrukken van
250 IF INKEY$="" THEN 250      — een willekeurige toets
260 J=USR0(0) MOD 20           — oproep subroutine
270 FOR L=0 TO J
280 N=USR0(0) MOD 6            — oproep subroutine
290 X=(N MOD 3)*80                                kopieert de
300 Y=(N \ 3)*100                                 ogen van de
310 COPY (X,Y)-STEP(60,60),1 TO (100,70)          dobbelsteen
,0                                                (van een tot zes)
320 BEEP:FOR W=0 TO 50:NEXT W                     op pagina 1,
330 NEXT L                                        overeenkomstig
340 GOTO 250                                      met de waarde
350 ' *** gegevens dobbelsteen ***                die door de
360 DATA 25,25                                    subroutine
370 DATA 25,10,25,40                              wordt
380 DATA 10,10,25,25,40,40                        teruggegeven
390 DATA 10,10,10,40,40,10,40,40
400 DATA 10,10,10,40,40,10,40,40,25,25            gegevens die de
410 DATA 10,10,10,40,40,10,40,40,10,25,4          plaats van de
0,25                                              ogen op de
                                                  dobbelsteen
                                                  bepalen
```

```
420 '** schrijf machinetaalsubroutine *
430 RESTORE 480
440 READ M$:IF M$="END" THEN RETURN
450 POKE AD,VAL("&H"+M$)
460 AD=AD+1:GOTO 440
470 ' *** machinetaal codes ***
480 DATA 21,F8,F7,ED,5F,77,23,AF
490 DATA 77,3E,02,32,63,F6,C9
500 DATA END
```

Voorbeeld van een subroutine in machinetaal

Het onderstaande programma in machinetaal dient voor het genereren van willekeurige getallen tussen 0 en 255, met behulp van het register R (het "refresh" register) van de CPU stuureenheid van de computer. De lijst hieronder is een zgn. bronnenlijst. (De schrijfwijze is gebaseerd op MACRO80 3.44).

```
        MACRO-80 3.44   09-Dec-81       PAGE    1

                                        .Z80
                                        .PHASE  0D000H

 0002                           INTEGER EQU     2
 F663                           VALTYPE EQU     0F663H
 F7F6                           DAC     EQU     0F7F6H

                                ;
                                ;--     machine-language sample program
                                ;

 D000                           START:
 D000    21 F7F8                        LD      HL,DAC+2
 D003    ED 5F                          LD      A,R             ; load R register
 D005    77                             LD      (HL),A
 D006    23                             INC     HL
 D007    AF                             XOR     A
 D008    77                             LD      (HL),A          ; set random data
 D009    3E 02                          LD      A,INTEGER
 D00B    32 F663                        LD      (VALTYPE),A     ; set data type
 D00E    C9                             RET

                                        END     START
        MACRO-80 3.44   09-Dec-81       PAGE    S


Macres:

Symbols:
F7F6    DAC             0002    INTEGER         D000    START
F663    VALTYPE



No Fatal error(s)
```

OPSLAAN VAN EEN MACHINETAAL-SUBROUTINE BSAVE

De subroutine in machinetaal die in het programma hierboven gebruikt is werd in de adressen &HD000—&HD00E in het geheugen geschreven.
Voor het opslaan van deze machinetaal-subroutine op cassette of diskette gebruiken we het BSAVE bevel.

BSAVE "[apparaatnaam] [bestandsnaam [.soortnaam]]", beginadres, eindadres [,beginadres verwerking]

Het BSAVE bevel dient voor opslaan van het gebied in het geheugen dat in het bevel is aangegeven.

```
BLOAD "CAS:RANDOM"
```

slaat de inhoud van geheugenadressen &HE000—&HE00E (het programma in machinetaal) op cassette op onder de bestandsnaam RANDOM.

```
BLOAD "A:RANDOM.BIN"
```

slaat dezelfde geheugeninhoud op op de diskette in diskette-eenheid A onder de bestands- en soortnaam RANDOM.BIN.

LADEN VAN EEN MACHINETAAL-SUBROUTINE BLOAD

De geheugeninhoud die is opgeslagen met het BSAVE bevel kan weer geladen worden met het BLOAD bevel.

BLOAD "[apparaatnaam] [bestandsnaam [.soortnaam]]" [,R] [,verschuiving]

```
BSAVE "CAS:RANDOM",&HD000,&HD00E
```

laadt de inhoud van het bestand dat onder de naam RANDOM op cassette was opgeslagen.

```
BSAVE "A:RANDOM.BIN",&HD000,&HD00E
```

laadt de inhoud van het bestand dat onder de bestands- en soortnaam RANDOM.BIN op de diskette in diskette-eenheid A was opgeslagen.

In beide gevallen wordt de inhoud geladen van het gebied dat loopt vanaf het in het BSAVE bevel aangegeven beginadres.

Als een subroutine in machinetaal op de bovenstaande manier afzonderlijk is opgeslagen, dan kan deze geladen en als onderdeel van een BASIC programma uitgevoerd worden zonder dat het nodig is de subroutine met een POKE bevel in het geheugen te schrijven. In het laatste BASIC programma zou dit dus regel 50 en de regels 420 t/m 500 overbodig maken.

INDEX

BASIC BEVELEN, INSTRUKTIES, FUNKTIES EN FOUTMELDINGEN

GEBRUIK VAN BASIC

Bediening

Maken van een programma

Verwerking van programma's en volgorde

Gegevens toewijzen en weergeven

Verwerken van numerieke gegevens

Verwerken van tekstgegevens

Laden en opslaan van programma's

Gegevensbestanden

Bestandsbeheer

Definities en instellingen

Bij gebruik van het tekstscherm

Bij gebruik van de grafische schermsoort

Kleur

Beeldpatronen

Programmeertechniek